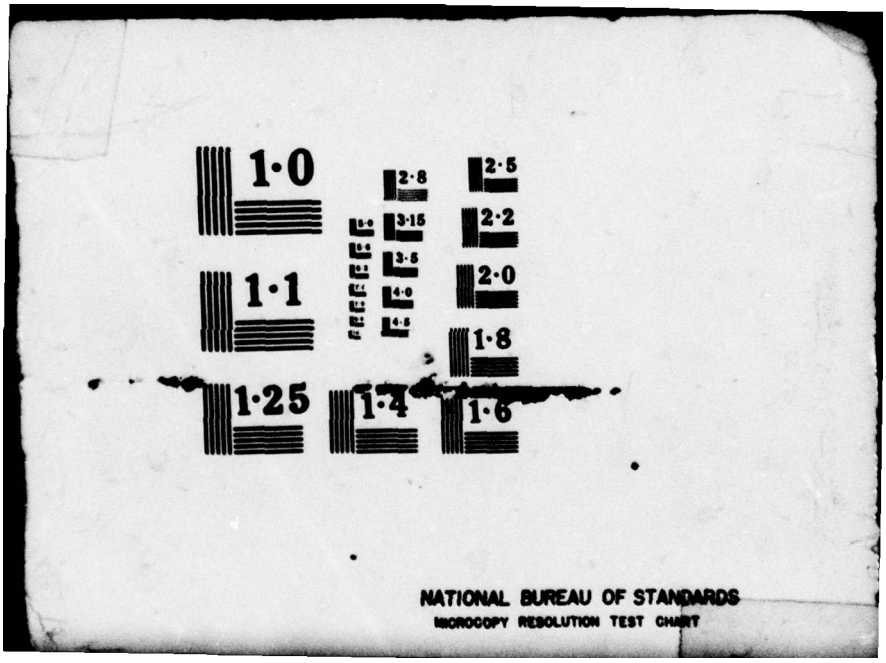
AD-A073 015 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL--ETC F/G 9/2
A PARAMETRIC MANAGEMENT TOOL FOR ESTIMATING SIMULATOR SOFTWARE --ETC(U)
JUN 79 G N FREY, K L WILDUNG

UNCLASSIFIED AFIT-LSSR-4-79A NL

1 OF 3
AD
A073015


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

AD A073015

DDC
AUG 24 1979
C

DDC FILE COPY

UNITED STATES AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

(3)

DDC
RECEIVED
AUG 24 1979
C

A PARAMETRIC MANAGEMENT TOOL FOR ESTIMATING SIMULATOR SOFTWARE SIZE

Gregory N. Frey, Captain, USAF
Kenneth L. Wildung, Captain, USAF

LSSR 4-79A

This document has been approved for public release and sale; its distribution is unlimited.

79 08 23 094

The contents of the document are technically accurate, and no sensitive items, detrimental ideas, or deleterious information are contained therein. Furthermore, the views expressed in the document are those of the author(s) and do not necessarily reflect the views of the School of Systems and Logistics, the Air University, the Air Training Command, the United States Air Force, or the Department of Defense.

USAF SCN 75-20B AFIT Control Number _____

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: AFIT/ LSH (Thesis Feedback), Wright-Patterson AFB, Ohio 45433.

1. Did this research contribute to a current Air Force project?

   a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

   a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of man-power and/or dollars?

   a. Man-years _____ $ _____ (Contract).

   b. Man-years _____ $ _____ (In-house).

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3 above), what is your estimate of its significance?

   a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments:

_____ _____

Name and Grade Position

_____ _____

Organization Location

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE. $300

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 73236 WASHINGTON D.C.

POSTAGE WILL BE PAID BY ADDRESSEE

AFIT/LSH (Thesis Feedback)
Wright-Patterson AFB OH 45433

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER LSSR 4-79A	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A PARAMETRIC MANAGEMENT TOOL FOR ESTIMATING SIMULATOR SOFTWARE SIZE		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Gregory N. Frey, Captain, USAF Kenneth L. Wildung, Captain, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Graduate Education Division School of Systems and Logistics Air Force Institute of Technology, WPAFB OH		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of Research and Administrative Management AFIT/LSGR, WPAFB OH 45433		12. REPORT DATE June 1979 13. NUMBER OF PAGES 188
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 199 p		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

JOSEPH P. HIPPS, Major, USAF
Director of Information

AFIT-LSSR-4-79A

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Base, Modular Software Design, Simulator, Sizing Model, Software

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Thesis Chairman: Lieutenant Colonel Donald R. Edwards

012 250

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This research investigated the causes of escalating software costs relative to Aircrew Training Devices (simulators) and resulted in a software sizing tool useful in reversing the cost growth trend. The hypothesis was that a collective software sizing estimate, aggregated and standardized at the functional level, would provide a more reliable estimate of simulator software size. Investigation disclosed that the heuristic sizing techniques being used were inadequate. There was no organized empirical data-base to support a formalized software sizing process. A software management model called SIMSIZ, including a supporting data-base, was designed and developed. The model's primary function is to assist software engineers in making realistic evaluations of the size of contractor's proposed software packages. It was concluded that a major cause of software program delays has been a growth trend in software size during program development. SIMSIZ has demonstrated that several software projects under development are undersized. The researchers recommend establishing a requirement to standardize simulator software development and reporting at the functional level.

Accession For
NTIS GRA&I
DDC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or special
A

LSSR 4-79A

A PARAMETRIC MANAGEMENT TOOL FOR ESTIMATING SIMULATOR SOFTWARE SIZE

A Thesis

Presented to the Faculty of the School of Systems and Logistics

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Logistics Management

By

Gregory N. Frey, BSES
Captain, USAF

Kenneth L. Wildung, BSBA
Captain, USAF

June 1979

Approved for public release;
distribution unlimited

This thesis, written by

Captain Gregory N. Frey

and

Captain Kenneth L. Wildung

has been accepted by the undersigned on behalf of the faculty of the School of Systems and Logistics in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN LOGISTICS MANAGEMENT
(ACQUISITION LOGISTICS MAJOR)
(Captain Gregory N. Frey)

MASTER OF SCIENCE IN LOGISTICS MANAGEMENT
(Captain Kenneth L. Wildung)

DATE: 13 June 1979

Donald R Edwards

ACKNOWLEDGMENTS

The authors wish to express their gratitude to the faculity and administration of the School of Systems and Logistics for their contribution to this thesis. Particular thanks goes to Lieutenant Colonel Donald R. Edwards for his positive guidance and the freedom to pursue our own interests in this research effort. To Howard Creek, fellow student and reader, many thanks for final proof reading.

We also want to acknowledge the investments made by the personnel of ENETC. Bob Cameron, chief of the Simulator Computer Branch, sponsored our research and provided technical support from his software engineering staff. Dave Butler, computer specialist, lent great assistance in data gathering, technical design, and guidance in the development of SIMSIZ. Clif Patterson, computer specialist, made significant contributions to our standardization of software module categories and assisted in exercising SIMSIZ.

Our final thanks goes to our wives, Kathy and Coleen, who deserve much more than thanks and usually receive far less. Their endless encouragement and intuitive understanding gave us strength to endure the ordeal. We are deeply indebted to Coleen for her extensive typing contributions and most certainly empathize with her many frustrations that are embodied in "RETRANSMIT LAST LINE."

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	iii
LIST OF TABLES	vi
Chapter	
I. INTRODUCTION	1
Overview of the Problem Area	1
Scope of the Research	4
Statement of the Problem	4
Background	5
Objectives	10
Research Hypothesis	11
II. METHODOLOGY	12
Introduction	12
Data and Data Sources	13
Data-Base Design	18
Model Design and Manipulation	19
General Model Verification	23
Meeting the Research Objectives	24
III. THE APPLICATION OF SIMSIZ	25
Data Base Verification and Update	25
Management Information System	26
A Software Package Sized	27

Chapter	Page
IV. CONCLUSIONS AND RECOMMENDATIONS	31
User's Evaluation of SIMSIZ	31
Research Conclusions	32
Recommended Changes to Data Reporting	34
Recommended Areas for Future Research	34
APPENDIXES	36
A. SIMSIZ USER'S MANUAL	37
B. DEFINITIONS OF STANDARDIZED MODULE CATEGORIES	79
C. GLOSSARY	85
D. F-15 VALIDATION SIZING PLAN	89
E. SIMSIZ SOURCE CODE	92
SELECTED BIBLIOGRAPHY	185
A. REFERENCES CITED	186
B. RELATED SOURCES	188

LIST OF TABLES

		Page
1.	Standardized Functional Module Categories	17
2.	Software Sizing Comparison	28
3.	SIMSIZ Validation Test Results	30

CHAPTER 1

INTRODUCTION

Overview of the Problem Area

The Air Force's acquisition and use of flight simulators has continued to increase in recent years owing primarily to the Department of Defense's (DoD) dedicated effort to reduce the rising costs of military flying time (18:2). Tighter federal budget constraints and the continuing rise in fuel costs since 1974 have fostered the DoD flying hour reduction effort. With the reduction in the flying hour program, flight simulators are receiving greater emphasis. The application of digital computers and sophisticated software has improved flight simulator fidelity.[1] A recent example is the addition of the Computer Generated Image (CGI) visual system to all of the C-5A and C-141A simulators. Operational Test and Evaluation (OT&E), conducted by one of the authors,[2] substantiated a significant increase in simulator acceptance by line assigned aircrews.

---

[1]Fidelity is the exactness with which the simulator duplicates aircraft performance and flight sensations.

[2]Captain Wildung conducted OT&E on the C-5 CGI visual modification while assigned to the Simulator Training Branch of the 60 MAW, Travis AFB, California.

Over the past four years there has been a dramatic increase in budgeted simulator funds: for fiscal year 1974 the budget provided $88.5 million for update and initial acquisition of simulators; for fiscal year 1976 the amount was $283 million (18:24). The projected five-year budget through fiscal year 1985 includes $1.8 billion for simulators (15). Expenditures of this magnitude require accurate budget estimates to avoid additional costs and program delays incurred when expenditures exceed original estimates.

Software cost estimates provide a basis for the financial support of software system development. If software costs are seriously understated, the system program manager must request supplemental appropriations. Inaccurate cost estimates are among the greatest contributors to underfunding (16:1). Requests for additional funds are quickly labeled as overruns and not as inaccurate estimates. To avoid these problems, system program managers need reliable techniques to estimate and budget for software development. Some general measures have been taken to improve simulator software cost estimates.

The Simulator Systems Program Office (SIMSPO) was created in May of 1973 to implement a standardized simulator acquisition strategy and to establish a central coordinating office for Air Force simulator technology and information (1:135). Software engineers assigned to AFSC's

Aeronautical Systems Division, Directorate of Engineering (ENETC), have been dedicated to the SIMSPO and are responsible for managing simulator software development. However, their software size estimates have been based solely upon heuristic techniques (3). They have no organized information source to support their suspicions that contractors deliberately underestimate memory size and timing requirements to buy-in on contracts. This inability to refute contractor proposed memory size and time requirements has resulted in extensive program delays. For instance, the C-5 Cockpit Procedural Trainer (CPT) is now estimated to be ready for training two years later than was originally proposed. A user originated Engineering Change Proposal (ECP) for inclusion of an inertial navigation package caused part of the program delay. However, the computer program manager estimated that at least a ten-month delay on the original proposal would have been inevitable (11). The C-130 Operational Flight Trainer (OFT), the C-130 CPT, and the C-130 visual system are expected to slip from five to twelve months because of insufficient spare processor time (4).

Recent improvements in simulator fidelity and realism, such as CGI visual systems, are attributed to the application of sophisticated computer software. The price of improved simulator realism is increasing software complexity. The production costs of simulator software are

estimated to be 85 percent labor intensive and are becoming a larger portion of total simulator costs (20).

Scope of the Research

In view of the magnitude and complexity of the software cost estimating problem, the scope of this research has been narrowed to the development of a parametric technique to forecast the size of simulator software packages. A more accurately sized software package, size being a major cost driver, will provide a more realistic basis for software costs (3). Software size, measured in terms of memory requirements and central processing time, are translated into the number of manhours required to produce the code (3). Contractor labor rates and overhead rates, projected over the acquisition process, are applied to the estimated manhours required to produce a software package. DoD-directed inflation adjustments are made to provide an aggregated software cost estimate (20). In summary, this research will take the parametric approach to forecasting simulator software size to provide an empirical basis for more reliable software cost estimates. Software cost estimation is not an objective of this research effort.

Statement of the Problem

The SIMSPO does not have an organized empirical data base to support a formalized software sizing process necessary for reliable cost estimating. Presently,

heuristic sizing techniques are based solely on the personal experience of software engineers. The fact that no parametric means exists for estimating software size has contributed to the inaccuracy of simulator acquisition cost estimates (15).

Background

A brief topical review of the history of software engineering[3] over this decade provides a background for the research approach.

The NATO Software Conference during 1969 provided the arena for international discussions of the costly and well-publicized software failures in operating systems. Software development costs were out of control. By 1971 the prevalent research was into fundamental programming practices. The advantages of top-down design, stepwise refinement, modularity, and programming team reviews gained special attention. The theme by 1973 shifted to structured programming with increased attention given to total software life cycle management. The International Conference on Reliable Software, in 1975, stressed reliability and quality assurance. Models designed with fault tolerances and total system reliability were the key areas of interest. Requirements, specifications, and design were the buzz words

---

[3]"A term coined in 1968 as a provocative term intended to highlight the need for a disciplined, process-oriented approach to software development [19:30]."

in 1976-1977 (19:31). Renewed attention was given to a thorough understanding of design requirements prior to coding. Much of the innovative software theory had been applied. Efforts were increased to integrate and validate succeeding development phases (8:69-84). It was time to reevaluate the basics of rigorous design and detailed specification requirements. By 1980 it seems likely that the emphasis will turn toward an increased use of automatic software development tools and a widespread use of the techniques and principles developed during the previous decade (17:81).

> As an engineering discipline, software engineering must concern itself more with modeling, experimenting with, and validating theoretical ideas, then using the results to build systems [19:36].

In spite of advanced techniques in software development, there remains a significant trend of rising computer software costs in all sectors of the computer industry. The DoD uses five basic types of computer systems, each of which suffers from software cost estimating problems (2:65). Each type will be briefly described prior to concentrating on simulator software problems.

1. Operational Flight Programs (OFP) refer to the software installed on board modern aircraft that comprise the basic components of computerized avionics systems.

2. Automatic Test Equipment (ATE) is the highly specialized hardware and software used by field, base, and depot level maintenance to perform automatic or

semiautomatic testing of operational systems such as complex aircraft avionics and missile guidance systems.

3. Communication Command and Control Systems (CCC) generally consist of large, high-speed computer systems located worldwide to provide near real-time information.

4. Automated Data Processing (ADP) systems are located throughout DoD to handle the day-to-day functions of the personnel, supply, and finance.

5. Aircraft Simulators use very sophisticated software and unique hardware to provide aircrew training under extremely realistic simulated flight conditions.

The symptoms of the software cost estimating problem are the excessive costs associated with development delays and substandard system performance which necessitate engineering changes to attain user operation requirements. Traditionally, software engineers and computer system program managers have emphasized quality control techniques in software development such as design reviews, software validation, debugging aids, and thorough testing (9:18).

These efforts have failed to cure escalating software costs. "Overruns of 100 percent in both cost and the time to develop software have not been unusual occurrences [9:19]." Software development costs are approaching 90 percent of total computer system costs (9:18). In 1973 it was projected that by 1985 the Air Force spending ratio on software versus hardware would grow to 10:1 (13:13). In

1977 the federal government estimated the annual cost of software development and maintenance at $4 billion (9:19). By 1978, the World Wide Military Command and Control System (WWMCCS) had already cost $722 million for software compared to less than $100 million for hardware (13:13).

Several potentially effective general purpose software cost models have been developed such as: the RCA PRICES model, the General Research Corporation model, and the Wolverton model. These parametric models are presently ineffective for simulator software cost estimates because the available data are either aggregated and reported at too general a level, or are altogether unavailable (1:17). Parametric cost estimating models, without a continuously updated data base, become obsolete and provide minimal assistance to the cost estimator (20). The potential of these models will not be realized until the detailed costs of software development are accounted for and reported by contractors.

Detailed software costs, until just recently, were not required to be furnished by contractual agreement (2:52). The rising software costs have now made it cost effective to purchase detailed data upon which future cost estimates can be refined. However, until an extensive empirical cost base can be established, alternative cost estimating techniques will have to be developed to alleviate the present budget-underfunding/cost-overrun problem.

The authors elected to concentrate on simulator software problems because of their mutual interest in this area and the likely prospect that simulator software would have a common denominator amenable to software cost estimating. This belief was based upon our combined experience in simulator operations, training, and modifications, and in the analysis and design of computer software systems.

A concentrated literature search of software studies provided only general information concerning flight simulator software. Several theses on various aspects of flight simulators were helpful and are included in the related sources of the bibliography. Our information search continued through interviews, which provided the conceptual basis for our research. ASD software engineers enumerated their current policies to combat the unpredictability of software costs: (1) FORTRAN, a standard High Order Language (HOL), is required, when feasible, for all software contracts; (2) software development contractors must use a structured top-down design containing functional modules; and (3) contractors are required by Air Force Regulation 800-14 to submit for review a computer program management planning and development plan for software development (2).

This research effort applies the present state of the art of software development to build a software sizing management tool. The size of a simulator software system depends largely on the amount of software modules and the

computer language used (1:21). The software modules are segmented by function into relatively standard subprograms, the type and size of which depends on the contractor's engineering design estimates of the complexities as specified by the Request For Proposal (RFP). A parametric sizing model could predict the size of the required modules needed to validate or refute the contractor's engineering estimates, provided an empirical data base was accessible. These data are available, but have not been aggregated into a usable form.

Objectives

The broad purpose of this research effort is to provide a parametric software sizing tool to assist in estimating simulator software acquisition costs. Our specific objectives to accomplish this purpose follow.

1. To gather empirical data on simulator software by type and size of modules, type aircraft, processing time, and memory size requirements. The data will include the full spectrum from Cockpit Procedures Trainers to the most sophisticated Weapon System Trainers.

2. To organize collected data into a workable data base.

3. To design and code an interactive computational software sizing model.

4. To verify and validate the software sizing model.

Research Hypothesis

The hypothesis of this study is that a collective software sizing estimate, aggregated and standardized at the functional level, can be used to provide a more reliable estimate of simulator software size than the present heuristic techniques being used. A more reliable software size estimate will reduce program delays and thereby reduce the software costs of simulator acquisitions.

CHAPTER II

METHODOLOGY

Introduction

SIMSIZ is the name given to an interactive computer model designed and developed by the researchers. It functions as a software sizing tool, a management information system (MIS), and provides a means for analyzing and updating its supporting data base. The primary purpose of the model is to give software managers an effective management tool for predicting the size of simulator software packages. The data base and SIMSIZ were designed to allow maximum flexibility in accessing and analyzing the information on simulator programs contained in the data base.

This chapter describes the nature and content of the source data collected and incorporated into the data base. The data-base design, which bears significantly on the capabilities of SIMSIZ, is covered in detail. The specific capabilities of SIMSIZ are addressed in concert with the model assumptions. Model validation, using internal and external tests, is explained. Finally, an examination of how well SIMSIZ satisfies the research objectives is discussed.

Data and Data Sources

The majority of the data used by SIMSIZ was provided by ASD software engineers dedicated to the SIMSPO. Part of this data consisted of the periodic (usually monthly) status reports required of various contractors of current simulator software development programs. These status reports generally contain the following information by module: (1) the module name or an alphanumeric designator, (2) the execution speed and frequency, (3) the memory requirements for instructions in either bytes or numbers of words,[4] (4) the computer language used (either FORTRAN or Assembly), and (5) the memory requirements for program data. The general lack of standardization, in the details of how this information is reported by the various contractors, generated several problems in comparative data analysis.

Some contractors report module size in "bytes" while others report in "number of words." A comparative analysis requires like units. A conversion factor from "number of words" to "bytes", for each of the computer systems used, is stored in the information section of the data base. SIMSIZ used this conversion factor to adjust the data records and standardize data output as "standard bytes." Data entries were made in the same format as reported by the respective

---

[4]Computer word size, usually an integer number of bytes, varies with the type of computer. A byte is a measure of storage capacity defined as eight bits. A bit is the smallest unit of storage and is assigned a value of one or zero.

contractors. This philosophy was elected, over standardizing the data prior to entry into the data base, for ease of data-base update and to reduce the chance of computational errors in the conversion.

An equally difficult problem with the data was that there is little commonality in the contractor's module naming conventions. Although most contractors organized their modules by major functional categories such as Aerodynamics, Engines, or Navaids, the detailed breakout of modules within these categories varied considerably from contractor to contractor. Even when two simulators were built by the same contractor, as in the case of the C-130 and the F-16, there are considerable variances in the functional designation of the individual modules within the major system categories.

Nonstandard nomenclature was also a major roadblock to the analytical comparison of available simulator sizing data. A set of standard module categories was devised to establish commonality among simulator programs. Individual module names for each of the included simulator programs were functionally grouped under the appropriate standardized categories, coded accordingly, and loaded into the data base.

The software engineers provided the contractor's technical engineering proposals for the simulator systems to be included in the data base. These proposals contained the

contractor's original estimates of software size, based on estimated central processor time and memory requirements, at the time their contracts were awarded. Unfortunately, the original proposals do not always break out estimates at a detailed modular level. This limits an analysis of software size growth to that level of detail given in the original technical proposals.

The modular nomenclature, as provided in the periodic status reports, was fairly consistent througnout any given simulator program. Contractors have occasionally changed the name of a particular module or added a few new modules to their software package. This was not generally a problem. Most contractors provided a four to eight character alphanumeric (alpha) designator for each module in addition to a module description.

The grouping of related modules for a particular category, while not always obvious from the module description, was made by reference to the alpha designator for the modules. For example, all modules which comprise an Inertial Navigation System (INS) would have alpha designators in the IN500 series (IN500, IN501, IN502, etc). While IN500 may not be a common notation among all contractors for referencing the INS subsystem, the general concept of grouping by alpha designator was found to be rather widespread.

This grouping concept was the key to establishing a standardized module categorization scheme across all contractors and types of simulators. The alpha designators of all modules for all simulators were grouped into functional categories listed in Table 1. Two-letter alphabetic codes were assigned to each function. Each module was coded by its appropriate functional category. This provided a cross-indexing capability by module which perform the same system functions for any of the simulators included in the data base. Table 1 shows the twenty-seven categories and respective codes standardized at the functional level. Appendix B contains definitions for each of the Functional Categories.

The other principal sources of data were the using commands of operational simulators. They provided data on the final values of core size for the software modules in their simulators. The intent for including this data was to provide the capability to analyze the growth in software size from contract award to full operational status.

The data on the F-15 simulator, one of the most recent simulators to become operational, were intentionally excluded from the original data base so that the F-15 could be used as a test case in the validation of the model. Once the validity of the overall model design had been established, the data on the F-15 were added to the data base via the SIMSIZ data-base Add subprogram.

TABLE 1

STANDARDIZED FUNCTIONAL MODULE CATEGORIES

Module Category	Code	Module Category	Code
Aural Cues	AC	Fuel System	FS
Auxiliary/External Power	AP	Growth Provisions	GP
Armament/Stores	AS	Hydraulics	HD
Avionics	AV	Instructional	IN
Cockpit Displays	CD	Landing Gear	LG
Communication	CM	Motion	MO
Computational Non-real	CN	Navigation	NV
Computational Real Time	CR	Radar	RD
Data Files	DR	Simulation Environment	SE
Engines	EG	Special Systems	SS
Electrical Power	EP	Tactics	TC
Environmental	EV	Visual Real	VR
Electronic Warfare	EW	Visual Support	VS
Flight Controls	FC		

Data-Base Design

SIMSIZ's secondary role is that of a Management Information System (MIS). The key to a successful management information system is the design of the system's data base. The data-base design was rigorously analyzed to insure proper format and the inclusion of pertinent data.

Each data record in the data base contains coded information by module by simulator. Each record is identifiable by the simulator type (i.e., F-5, C-130, etc.), year of latest record update, major system function, subsystem function, module description, and module alpha designator. Each record also contains the contractor's original estimate, if available, and the latest/final values for module central processor time and memory requirements. The records also include Iterations per Second (IPS), which tell how often a particular module is executed, and a code for the computer language in which the module is written. In order to facilitate data-base file maintenance, each record also contains a delete mark which is set if the user wishes to delete a particular data record entry completely. Once the delete mark is set, the record is ignored in all future data retrieval tasks. The data are not immediately removed from the file but remain until there are sufficient deleted records to cause the data-base file to be repacked, at which time the record is irretrievably lost.

The design, development, and testing of the data base and the SIMSIZ subprograms were conducted on the Air Force Logistics Command's (AFLC) CREATE computer system. The SIMSIZ software was written in Honeywell FORTRAN IV and is listed in Appendix E of this thesis. The details of the data-base design including the control values, directory information, and data record contents are included in the SIMSIZ User's Manual contained in Appendix A.

The data assumptions are as follows:

1. That the data were reported accurately on the source documents.

2. That the contractor's latest estimates of module sizes on unfinished software packages represent a reasonable estimate of the final module sizes.

3. That the grouping of modules into the standardized functional categories by the respective software program managers and the researchers accurately reflect the function of the modules.

4. That the transfer of data from source documents to the data base was accurately accomplished.

Model Design and Manipulation

SIMSIZ is an effective multipurpose computer model that assists in software size estimates, provides the user with a data-base update/delete capability, and functions as a Management Information System (MIS).

SIMSIZ was specifically designed to overcome encroaching obsolescence. Each of the data values for a module or group of modules, contained in the data base, may be adjusted by a weighting factor and a complexity factor. For example, the data for the C-5 simulator reflect an earlier software production technology than does the data for the C-130. It might be desirable to give greater weight to the C-130 data than to the C-5 data in order to cause the size estimate to favor the later software production technology. A complexity factor can also be used to inflate or to reduce the aggregated size estimate for a particular functional category or for an entire simulator package. The relative complexity factor is interactively provided to SIMSIZ by the designing software engineer. Guidance concerning the use of weighting factors and complexity factors is provided in Section VII of the SIMSIZ User's Manual (see Appendix A).

A SIMSIZ user can more accurately predict software package size. It allows him to scan the module categories in the data-base directory and to identify each module data-record in the data base which he wishes to include in his new simulator design. Once all appropriate modules have been selected and marked, a conventional sizing run can be initiated; only the marked modules will be included in the computations. The modules selected from the data base will

retain their mark for later access should the designer desire to modify his proposed software package.

A conventional sizing run is an accounting function which aggregates totals from the actual timing and sizing data contained in each user selected module. These data may be modified by the user by applying adjustment factors which he supplies during the run. Section VII of Appendix A contains the equations which show how the conversion factors for computer speed, computer word size, and programming language are applied to the module data.

The designer may also use an engineering estimated value for a particular module. The use of this feature is envisioned for the design of functional subsystems not previously performed by simulators such as launch, monitoring, and scoring of cruise missiles.

SIMSIZ can extract information sorted on any key field or any combination of key fields maintained within each data-record (i.e., type simulator, year of latest data entry, major system function, subsystem function, or alpha designator). This versatile feature underscores the effectiveness of SIMSIZ as a MIS. For instance, if the manager wishes to know the average size of all the modules in system Navigation, subsystem TACAN, for fighter type simulators representing software technology of the past two years, such information could be supplied by module description and

alpha designator. See Section IV of Appendix A for instructions on the Select function of SIMSIZ.

In addition to the software sizing and MIS features, SIMSIZ provides the user with the capability to update or modify existing data-records. New information can be added to the data-base either by adding a new module to an existing simulator package or by adding a new simulator package. Complete instructions are contained in Section III of the SIMSIZ User's Manual.

Our intent was to make the system as flexible and as easy to maintain as possible. However, a full understanding of data-base maintenance can only be gained by studying the SIMSIZ User's Manual and by exercising the system. In SIMSIZ's role as a software sizing model, the interactive sequences are sufficiently descriptive that an occasional user can design a simulator software system with minimal reference to the user's manual.

The SIMSIZ model assumptions are as follows:

1. There have been no major technological breakthroughs in computer software production that would significantly reduce the amount of program instructions needed to perform a given function. Minor improvements in coding efficiency can be compensated for with weighting factors or complexity reduction factors.

2. The update/delete capability of SIMSIZ will be exercised to insure that current data from on-going and new simulator contracts are contained in the data base.

General Model Verification

Model verification was accomplished in three phases. The first phase consisted of internal tests of the data base and the SIMSIZ data manipulation routines. The second phase tested the software sizing feature. The update/delete routines that modify the data base were exercised in the final phase of verification.

Phase-one testing was conducted at the subprogram level. Each subprogram in the SIMSIZ software package was tested individually for functional validity, positive error recovery, logical accuracy, and the maintenance of data-base integrity. The manipulation routines were used to modify and update errors discovered in the data base during the line-by-line data-base audit. Data-base integrity, after the modifications and updates were completed, was verified against source documents and data base format requirements. The successful completion of these tests constituted verification of the manipulation subroutines.

Phase-two testing consisted of an evaluation of the software sizing feature. SIMSIZ was used to rebuild each of the simulator software packages contained in the data base. Each rebuilt simulator package was compared to the source documents and found to be identical. This procedure

verified the integrity of the three Run options of SIMSIZ (see Appendix A, Section VII for an explaination of the Run options).

Once the sizing tests had been successfully accomplished, the final phase of model verification was undertaken. This phase tested the data base update/delete routines used for file modification and maintenance. The most current data changes to the simulator software programs still in development were entered into the SIMSIZ data base. The correct operation of the modification routines was verified through examination of the data base extracted with utility dump routines (see Appendix A, Section VIII for an explaination of utility dump routines). The data-base listings were examined and found accurate; this concluded the three-phased model verification procedure.

Meeting the Research Objectives

The four specific research objectives were satisfied by the successful completion of the SIMSIZ validation tests. The broad objective of the research was to provide a parametric software sizing tool to assist in estimating simulator software acquisition costs. The output from SIMSIZ can be used to manually calculate simulator software acquisition costs; it could also be used as inputs to one of several software cost models discussed earlier.

CHAPTER III

THE APPLICATION OF SIMSIZ

Data Base Verification and Update

The Build function of SIMSIZ was used to initialize the data base, sequential permfiles[5] were created to temporarily store the data records for each of the simulator programs. Listings of the data records on the permfiles were compared, record-by-record, with the source documents. This line-by-line audit comprised the data record verification of the unformatted records on the permfiles. The Add function of SIMSIZ was used to retrieve the unformatted data records from the permfiles and to add them to the data base in the proper format. The validity of the Add function was determined by successfully completing the data retrieval and data-base update tasks.

Data verification at the aggregate level was accomplished using several SIMSIZ functions. The select function was used to make separate selections of each simulator program contained in the data base. Each selection was written to the TEMP file and then transferred to a permfile

---

[5]Permfiles are permanent files residing on disk space allocated to the user. Once created, the file remains until the user releases the file by name.

using the SIMSIZ PUT function. A hard copy of the permfile contents was then obtained using utility dump routines.

The printouts of all data-base records for each simulator were reviewed for commonality with the source documents (this verified the Add routine) and given to their respective program managers for evaluation.

The errors and omissions discovered during the first data-base verification process were corrected on the source data permfiles. The Build and the Add functions were used to reinitialize the data base and to write the revised data records onto the data base. The Modify function could have been used to update the original data base. However, due to the number of changes required, it was easier to simply rebuild the data base.

A similar verification process was performed on the revised data base. Minor errors were corrected using the SIMSIZ Modify function.

Management Information System

The use of SIMSIZ as a Management Information System (MIS) is limited only by the user's imagination and the content of the data base. Some of the MIS capabilities envisioned by the designers include: to give software engineers and management quick access to the size and timing requirements of all simulator programs, to make comparative analysis among contractors, to analyze the effects of using different computers on a given software program, to predict

the effects of allowing a contractor to change computer languages, and to provide a tool that management can use to monitor the growth trend in software size and timing. Table 2 reflects the growth trend in software size of some of the simulator systems recently completed or still in development.

Software management personnel can use SIMSIZ to monitor program progress in the temporary absence of their software engineers. With the high turnover of software engineers, the transfer of program management responsibility is enhanced by the standardized handling of data and software program information.

A comparative analysis of historical performances by contractors provides useful information in evaluating future contract proposals. This might include their selections of computers, their history of requesting waivers to coding programs in FORTRAN, and the historical growth rate of their previous simulator contracts. The essence of using SIMSIZ in a MIS capacity is to have the right information readily accessible to the right people at the right time.

A Software Package Sized

SIMSIZ was designed as a software sizing tool. The skill and experience of the user is the primary factor in developing a realistic size estimate. A software engineer, given sufficient time to consult aircraft technical manuals

TABLE 2

SOFTWARE SIZING COMPARISON

Trainer Type	Manufacturer's Original[1] Estimate			Manufacturer's current[2] Estimate		
	Instructions (bytes)	Data (bytes)	Timing (msec/sec)	Instructions (bytes)	Data (bytes)	Timing (msec/sec)
C-130 OFT	88,454	22,027	1,943.50	1,074,306	456,961	4,867.68
C-141 CPT	95,000	70,040	553.00	122,942	53,880	531.00
C-5 CPT	33,543	11,045	382.40	150,856	24,425	532.00
A-10 OFT	104,541	243,460	1,140.00	836,872	217,280	1,753.55
F-16 OFT	421,859	231,195	2,368.00	437,960	230,616	2,140.66
F-15 OFT[3]	325,774	232,277	1,849.06	492,105	222,675	3,251.70

[1]Values taken from contractors' original proposals provided by the ENETC.

[2]Values, as of 1 June 1979, that were taken from documentation provided by the ENETC (6).

[3]Values represent actual size of operational simulator systems.

and RFP specificatins, could design a highly detailed and realistically sized software package using SIMSIZ.

The conceptual basis of the model is that similar functions of similar systems generally require a similar amount of computer instructions, data storage, and central processor time. The validity of this assumption was demonstrated by redesigning an Operational Flight Trainer (OFT) based upon data from similar trainers.

The F-15 simulator was selected to exercise the predictive feature of SIMSIZ for several reasons: it was the most recent OFT to become operational, the F-15 contractor's original proposal was available, data on the actual size and timing values at the Ready for Training Date was acquired.

The validation test using the F-15 was designed to evaluate the use of SIMSIZ under the least favorable conditions (inexperienced user and a short time suspense).

Cliff Patterson, an ENETC[6] software engineer, agreed to design a software package to test the use of SIMSIZ. Mr. Patterson, a relatively new employee of ENETC, was chosen because he was unfamiliar with the details of the F-15 program. He was given the contractor's proposal for the F-15 simulator. This scenario simulated receiving a short suspense request to evaluate a contractor's proposal for a

---

[6]ENETC is the office symbol for the ASD software engineers assigned to the Directorate of Engineering.

new Operational Flight Trainer. He was given two days to become familiar with the F-15, to design his plan for sizing, and to exercise SIMSIZ to predict the software size and timing requirements for the F-15. Table 3 summarizes the size estimate[7] of his SIMSIZ run. Values for the contractor's original proposal and the F-15's actual operational size are included for comparison.

TABLE 3

SIMSIZ VALIDATION TEST RESULTS

Comparison Values	Instructions (bytes)	Data (bytes)	Timing (msec/sec)
Original estimate[1]	325,774	232,277	1,849.06
SIMSIZ estimate	592,605	230,375	2,053.65
Actual size[2]	492,105	222,675	3,251.70
Original as a % of actual	66.2 %	104.3 %	56.9 %
SIMSIZ as a % of actual	120.4 %	103.5 %	63.2 %

[1]Values were taken from the contractor's original proposal provided by ENETC(6).

[2]Values were taken from contract documentation provided by ENETC(6).

---

[7]See Appendix D for Mr. Patterson's sizing plan developed at the functional module level.

SECTION IV

CONCLUSIONS AND RECOMMENDATIONS

User's Evaluation of SIMSIZ

The primary users of SIMSIZ are the software engineers in ASD/ENETC, attached to the SIMSPO. Bob Cameron, branch chief of ENETC, assigned one of his software engineers, Dave Butler, to assist in data gathering, conceptual model design, and Operational Test and Evaluation (OT&E). Mr. Cameron summarized his assessment of the model with the following comments.

> SIMSIZ will be very useful to the design engineers as a software sizing and evaluation tool. Additionally, it is the basis upon which I intend to expand our effort to computerize information seriously needed for management of software development. Its use as a MIS will be invaluable. It provides easy access to the current status of simulator programs on contract. In this capacity it is especially valuable for maintaining management continuity during the temporary absences of the software program managers. SIMSIZ's flexibility in data-base management more than anything will insure its continued use and further refinement. I have applied for increased manning under the Student-Summer Aid Program to exercise SIMSIZ and to develop its potential [6].

A more technical evaluation was made by Mr. Butler during the OT&E phase.

> The sizing validation tests made on the F-15 proposal, though very supportive of the conceptual design of SIMSIZ, do not accurately measure the validity or potential of the model. Two variables within the model, deemed important to a valid output, are currently using default values of 1.0: the CPU Timing Conversion Factor and the computer language conversion factor.

ENETC personnel have arranged for the necessary values to be provided by the simulator manufacturers for later inclusion in SIMSIZ. As a general software sizing tool, SIMSIZ reduces design time to a third of what it has been and makes available ten times the information a designer normally has upon which to base his engineering estimate. The numerical validity of the model is dependent upon the experience of the user and the refinement of the aforementioned variables. The conceptual validity was unquestionably established by the short-suspense F-15 OT&E scenario.

The benefits derived from this research are numerous. The Standardized Module Categories developed are being recommended to contractors as a preliminary measure toward a mandatory standard for development and reporting. Our data on all available simulator programs has been organized into the standard module categories and computerized. We have been given an extremely flexible software sizing tool that doubles as a data-base manager, and a management information system. The addition of the CPU and language variables will culminate an excellent piece of applied research [5].

Research Conclusions

Information gained during the design, development, and operation of SIMSIZ, provided the basis for the following five conclusions.

1. The status reports of simulators under development at the time of this research provided sufficient descriptive detail to logically organize most of the software modules into the Standardized Functional Categories. It is conceded that a few of the modules probably performed functions in two or more of the standard module categories. If so, this would degrade the accuracy of the conclusion derived from the data base. However, sufficient consistency in reporting enabled the researchers, with the support of the respective software program

managers, to construct a comparatively reliable data base upon which the second conclusion was based.

2. Similar categories of simulators, such as CPTs and OFTs, require similar amounts of computer instructions, data storage, and CPU time to simulate basic aircraft functions. The degree of similarity is dependent upon the accuracy of the data base with respect to the placement of modules into the Standardized Functional Categories. The basic aircraft functions considered very reliable include the following Standardized Functional Categories: Auxiliary/External Power, Communications, Engines, Electrical Power, Environmental, Flight Controls, Fuel Systems, Hydraulics, Landing Gear, Motion, and Radar. The remaining categories were suspected of some crossover into other functional areas. However, at the simulator system level, the data base's accuracy was verified.

3. The high reliability of the data base at the system level substantiated the conclusion that a collective software size estimate aggregated from a functional level will provide a realistic estimate of simulator software size.

4. The size of software programs tend to expand during development. The cause is usually attributed to user generated ECPs. Although the evidence was not conclusive, sufficient instances of growth were revealed, in which ECPs were not a factor, to suggest that contractors tend to

underestimate the amount of software required to provide the specified simulation tasks.

5. SIMSIZ is a reliable software management tool that can be used to improve software size estimates, as a Management Information System, and as a data-base management system.

Recommended Changes to Data Reporting

This research disclosed that there was very little consistency in reported data among the various simulator programs. A consistent scheme of data reporting must be established to provide the comparability necessary for effective software management decisions. The recommendation of this research is to use the established Standardized Functional Module Categories, defined in Appendix B, as the standard level for simulator software development and reporting. Design flexibility within the defined functional levels should be given to the contractors to facilitate innovative software designs which simulate the standard aircraft functions. Development and reporting standards should be contained in the RFPs upon which manufacturers base their bids for contract.

Recommended Areas for Future Research

The broad objective of this research was to provide a parametric software sizing tool to assist in estimating simulator software acquisition costs. This was

accomplished, however several factors in determining reliable estimates of acquisition costs remain to be studied.

Research is needed in the area of translating software size into manhours required to code the software packages. Contractor labor rates and overhead rates are required to generate realistic software cost estimates.

A second area badly in need of research is simulator life cycle costs (LCC). SIMSIZ provides the capability for reliable software size estimates, which is one of the more sensitive variables for life cycle costs. An effort to interface SIMSIZ with some of the available LCC models would make an interesting and highly useful research effort.

APPENDIXES

APPENDIX A

SIMSIZ USER'S MANUAL

TABLE OF CONTENTS

Section		Page
I.	GENERAL INFORMATION	41
	Introduction	41
	How to Log-On to CREATE and SIMSIZ	43
	How to Obtain Instructions for SIMSIZ	45
	Responding to Questions Asked by SIMSIZ	45
	Permfiles and Tempfiles	46
	How to Log-Off of SIMSIZ and CREATE	47
II.	INITIALIZING A DATA BASE	49
	How to Allocate/Release File-space under CREATE	49
	Running the SIMSIZ Build Function	51
III.	ADDING TO THE DATA BASE	53
	Formatted vs Unformatted Records	53
	Adding Data from Files	53
	Tempfiles	53
	Permfiles	54
	Adding Data from the Terminal	54
	Test vs Real Add Function	56
IV.	SELECTING/LISTING RECORDS FROM THE DATA BASE	58
	Selecting Records from the Data Base	58

Section Page

Listing Records 61

From the Data Base 61

From the SIMSIZ TEMP File 62

V. MODIFYING/UPDATING THE DATA BASE 63

Data Records 63

Marking, Changing, or Deleting Selected Records 63

Updating the Data Base with Selected Records 63

Directory Section 64

Adding, Changing, or Deleting Entries 64

Information Section 66

CPU and Language Information 66

Simulator System Information 67

VI. SAVING/RETRIEVING DATA RECORDS ON PERMFILES . . . 70

How to Use the SIMSIZ Put Function 70

How to Use the SIMSIZ Get Function 70

VII. USING THE SIMSIZ SIZING FUNCTION 72

Option 1 - A Sizing Run Using the Entire Data Base 72

Option 2 - A Sizing Run Using the SIMSIZ TEMP File 73

Option 3 - A Sizing Run Using Selected Data Base Records 74

Complexity Factor Option 74

Weighting Factor Option 74

New Record Option 75

Section Page

Section Page

VIII. UTILITY ROUTINES FOR OBTAINING LISTINGS 77

Listing an Entire Data Base 78

Listing Formatted Permfiles 78

SECTION I

GENERAL INFORMATION

Introduction

This user's manual has been developed as a quick-reference guide to the capabilities and limitations of SIMSIZ. SIMSIZ is a collection of computer programs maintained on the CREATE computer system. The manual assumes that the user is somewhat familiar with data processing concepts such as: time-sharing operations, file space allocation, sequential and random access files, data fields and records, and modularized software packages. It also assumes that the user is familiar with the concept of sizing a software package using the Standardized Functional Module categories defined in Appendix B of the thesis.[1]

The user must access and execute SIMSIZ using a time-sharing system (TSS) terminal; outputs are designed for a TSS printer, with the exception of utility dump routines explained in Section VIII. It is recommended that a terminal with a hard copy printer be used for subsequent review/analysis.

---

[1]Frey, Captain Gregory N., USAF, and Captain Kenneth L. Wildung, USAF, "A Parametric Management Tool For Estimating Simulator Software Size," Unpublished master's thesis, LSSR 4-79, AFIT/LSG, Wright-Patterson AFB, OH, June 1979.

A typical software sizing plan for a user unfamiliar with SIMSIZ follows:

1. Review all of Section I of this manual.

2. Create permfiles if necessary, in accordance with Sections I, II, and III of this manual.

3. Determine which functional categories, defined in Appendix B of the thesis, are required by the Proposed Software Package (PSP).

4. Develop a sizing plan, by function code, around existing data available through the use of the Select routine (see Section IV).

5. Select one of the three Run options, explained in Section VII, and proceed with the instructions for the selected option. Option one provides for sizing using all the data in the data base. Option two is similar to option one except that the user can run against particular data which he has chosen with the select routines. Option three allows for complexity and weighting factor adjustments to the records. In addition, new records can be created and stored in the data base for future analysis/refinement.

6. Use the results of the sizing run (or multiple sizing runs) to refine the PSP to the user's satisfaction.

The remainder of the sections in this manual provide instructions on data base management and the use of SIMSIZ in obtaining management information. The respective section titles describe their respective functions.

How to Log-On to CREATE and SIMSIZ

It is assumed that the user is familiar with logging on to the CREATE system to include dialing the correct telephone number and connecting the terminal, through the modem, to the telephone line. Throughout this user's manual the required user responses to system inquiries are underlined. A carriage return (push the Return key) is required to terminate an entry. The following is a sample CREATE/SIMSIZ Log-On sequence.

STATION ID-PA (PB, PC, etc., can be used)

user id -SIMCOM

password-[illegible]

problem no.-[illegible]

SYSTEM? FORT N

READY
*

At this point the user has Logged-On to CREATE and is at what is termed the 'star' (*) level. He must now start the SIMSIZ program.

RUN SIMSIZ

ENTER CAT/FILE STRING OF DATA BASE FILE, FOLLOWED BY ';'.

=DATABASE;

WHAT IS THE MAXIMUM NUMBER OF CHARACTERS/LINE YOUR TERMINAL CAN PRINT?

=80

(This response determines the format used by SIMSIZ to generate listings of data records. Any number less than 120

causes the output to be formatted for an 80-column device. If the number is greater than or equal to 120 the output is formatted for 132 columns.)

HAS THIS FILE BEEN INITIALIZED (Y OR N)?

=Y

(If the answer to this question is 'Y', SIMSIZ reads certain sectors of the data-base file and loads header information into the COMMON areas used by the SIMSIZ routines. Answering 'Y' implies that the data-base file has been initialized by the SIMSIZ Build function (see Section II). Normally the correct response here would be 'Y'. When building a new data base, not yet initialized by the Build function, the response MUST be 'N'. A 'Y' response will abort the run.)

WELCOME TO PROGRAM SIMSIZ. WOULD YOU LIKE INSTRUCTIONS (Y OR N)?

=N

(See the next section on obtaining instructions from SIMSIZ.)

FUNCTION?

=

(Respond to "FUNCTION?" with your desired SIMSIZ function.)

How to Obtain Instructions for SIMSIZ

Each of the major functions performed by SIMSIZ (adding records, selecting records, modifying records, etc.) have abbreviated instructions contained within their respective programs. There are several ways of obtaining instructions.

After each log-on, SIMSIZ will ask the user if he wants instructions. A 'Y' answer will receive general instructions on all SIMSIZ functions. When selecting a particular function (for example 'A' for Add) the Add subroutine asks if the user wants detailed instructions. This will occur each time he initiates a new function. The process will continue until an 'N' response to the 'Instructions?' question is given. After an 'N' response, the user will no longer be asked whether he wants instructions. (Note: Responding with an 'N' to the 'Instructions?' question during Log-On, will prevent the question from being asked again.)

A user can return to the "Instructions?" mode by responding 'I' to the "FUNCTION?" question. In the case of the 'R' or RUN function the response to obtain instructions is '4'.

Responding to Questions Asked by SIMSIZ

When SIMSIZ expects an input from the terminal, it will prompt the user by typing an equal sign (=) in column one. SIMSIZ types the question on one line and the '=' on

the next line. Occasionally, when the CREATE system is very busy, there may be an appreciable delay between the typing of the question and the typing of the '=' prompting character. A response entered before the '=' is typed will be ignored by the system. Always wait until the '=' has been typed before entering your response.

Permfiles and Tempfiles

There are two major types of files which can be created and used on the CREATE computer system; permfiles (permanent files - which remain in the system until they are released by the user) and tempfiles (temporary files - which are released automatically by the system when the user logs-off or when the system goes down for repairs). The SIMSIZ data-base file is a permfile and any files which a user creates will also be permfiles.

Some functions of SIMSIZ require additional tempfile space to temporarily hold selected data records. SIMSIZ automatically requests needed tempfile space from the CREATE operating system. The SIMSIZ documentation refers to this space as the SIMSIZ 'TEMP' file. Anything written to this file during execution is lost when the user logs-off SIMSIZ. Therefore, if there is information on the TEMP file that the user wishes to retain, it must be transferred to a

permfile (by using the SIMSIZ PUT function) before the user logs-off.

Permfiles cannot be created by SIMSIZ. Any permfiles that the user anticipates a need for during a SIMSIZ run must be created before initiating the SIMSIZ program. See Section II of this user's manual for detailed information and examples of how to create/release permfiles.

How to Log-Off of SIMSIZ and CREATE.

The recommended method of terminating a SIMSIZ run is to enter 'E', for End, at the FUNCTION? level. Pushing the 'break' key on the terminal will not terminate SIMSIZ; the program returns immediately to the FUNCTION? level. Pushing the break key is an approved method of prematurely terminating most SIMSIZ functions (List, Run, Get, etc.) and will normally not cause any problems or disruptions to the data base. However, the user is cautioned that pushing 'break' while SIMSIZ is performing a function that writes to the data base (for example, the Add or Update functions), will prematurely terminate (abort) that function and could result in a disparity between the actual data on the data-base file, and what the directory and header sections say is on the file. A good rule is: don't use the 'break' key to terminate a function unless it is an innocuous function like List, Select, Run, or Get.

Each function, when completed, will either return automatically to the FUNCTION? level or allow the user to return at his discretion, by entering 'E'.

Once the user has entered 'E' at the FUNCTION? level, SIMSIZ will write certain data (from the COMMONS) back to the data base, de-allocate the data-base file, and terminate its own execution. The user will then be prompted by the CREATE operating system with the '*' character. Type 'BYE' to Log-Off the Create system when prompted by the '*'.

SECTION II

INITIALIZING A DATA BASE

How to Allocate/Release File-space under CREATE

All files in the CREATE system are either random access or sequential access files. The files that are used as data-base files must be created as random access permfiles. Files created as holding files or as source data files must be sequential access files. The only files that you as a user will create are sequential access permfiles.

The standard unit of file size, when dealing with the CREATE system, is a Block or Little Link (LLINK). Each LLINK is 320 words in size (a word is 36 bits long). A file 10 LLINKS in size can hold 3,200 words of data.

The minimum recommended size of a data-base file is 10 LLINKS. All data-base files require eight sectors for overhead. A 10 LLINK file would have room for approximately 170 data records.

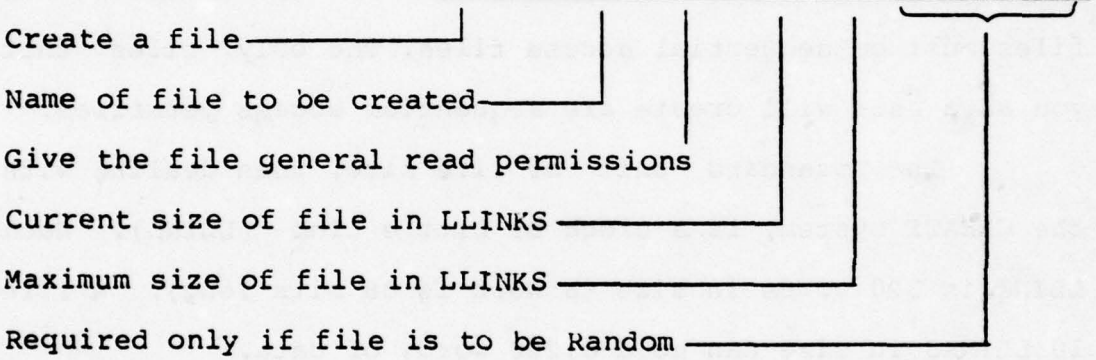
Sequential access (or "Linked" files) may be created at one size, with the added specification as to the maximum size to which the file may "grow". When a user creates a sequential permfile he may set the current size at 1 LLINK and the maximum size at 10 LLINKS. When the 1 LLINK becomes full, the file space will automatically grow, in 1 LLINK

increments, to accommodate additional data. The maximum size, in this example, has been set by the user at 10 LLINKS. The file will not grow beyond that size.

Before attempting to create a permfile or build a new data base using SIMSIZ, a user must first create a random access permfile on the CREATE system. This is done as follows: Log-On to the CREATE system as explained in Section I. When the system responds with "*" enter:

ACCESS

FUNCTION? CF,/Filename,R,B/nnn,nnn/,MODE/RAND/



The file name may be up to 10 characters; no embedded spaces. For random access files current size must equal Max size. All values should be entered as indicated, using slashes (/) and commas as shown, with no spaces between characters. It is not necessary to specify "MODE/SEQ/" for sequential files since SEQ is the default mode.

If your USER ID has sufficient file space available to create the file at the "current" requested size and if there are no syntax or other errors (duplicate file name,

current size greater than Max size, etc.), file space will be allocated as specified. The system will respond with:

SUCCESSFUL.

FUNCTION?

The system returns to the FUNCTION? level to offer the user a chance to create another file. The user may return to the "*" level by pushing the return key.

If new files are created indiscriminately you will eventually run out of filespace allotted to your USER ID. A policy of purging outdated files will provide room for new files. Alternatively you may simply overwrite an old file with new data. Files may be released at the "*" level by entering:

RELEASE Filename

(The "Filename" must be the exact name of the file to be released.)

The system will respond with:

FILE RELEASED-Filename

The minimum recommended size of a data-base file is 10 LLINKS. All data-base files require eight sectors for overhead A 10 LLINK file would have room for approximately 170 data records.

Running the SIMSIZ Build Function

After the data-base file space has been created/ allocated, as described in the previous paragraph, the file space is initialized using the Build (B) function of SIMSIZ.

The following steps must be performed in sequence to accomplish the build task.

1. Log-on as explained in section I.

2. Enter the filename of the new data-base file in response to the "ENTER CAT/FILE STRING . . . " question. (Both the USER ID and filename are required only if using filespace on another USER ID).

3. Give the terminal carriage size.

4. Important-- Answer 'N' to the question "HAS THIS FILE BEEN INITIALIZED?"

5. Enter 'B' to the FUNCTION? question.

6. Enter the name of the new data-base file when asked.

7. Give the size of the data-base file you created.

8. Answer questions as appropriate until the build task is completed and the program returns to the FUNCTION? level.

SECTION III

ADDING TO THE DATA BASE

Formatted vs Unformatted Records

The first question asked by the Add function is, "Is the data for the records, to be added, formatted or unformatted?" Formatted data are data that are in the same form as a standard SIMSIZ data record. Data that have passed through the TEMP file are formatted. For example, the user would have added "formatted" data if he had taken existing data-base records, modified them, and added them back to the original data base. When adding records via the terminal, they are "unformatted" because the fifteen fields of each record can be entered in any order. Likewise, data placed on a user sequential permfile using the CREATE "TEXT EDITOR" function are considered "unformatted" data.

Adding Data from Files

Tempfiles

To add data from the SIMSIZ TEMP file to the data base, give "TEMP" as the name of the file containing the data to be added. Data on the TEMP file are "formatted" data and will be added even if the user answered 'U' to the formating question. Delete marked records will not be added to the data base.

Permfiles

Formatted data records which have been placed on a user's sequential permfile (via the SIMSIZ "PUT" function), may be read from the permfile and added directly to the data base using the Add function.

Permfiles may also contain unformatted data which the Add function reads just as though they were being entered via the terminal. This option is useful if the user has a large number of new records to be added to the data base and does not wish to enter the records one at a time. Obtain a listing (using the Utility Dump routine) and verify the data on the permfile prior to adding the data directly to the data base. Unformatted data on a permfile must appear exactly the same as it would when entered from a terminal with one important exception. The first line of an unformatted data file must contain a series of numbers, separated by commas, which explain the order in which the data record fields are being entered. Fields for which there are no data can be omitted. The Add program resequences the data fields prior to adding the data to the data base.

Adding Data from the Terminal

Table A contains a description of the various data-record format specifications. The first fifteen may be specified by the user when creating a new record. Any or all of these fields may be specified for each record

entered. At the beginning of the run the user specifies the order, by field number, in which the fields will be entered. For example:

= 1,3,10,5,7,2,6,8,4,9

would mean that each line of data entered from the terminal will contain fields 1,3,10,5,7,etc., in that order. (The Add function rearranges the fields into the proper order for data base record format.)

TABLE A

DATA-RECORD FORMAT

FIELD NUMBER	FIELD FORMAT	DISCRIPTION
1	A1	Simulator Code
2	A2	Function Code
3	A2	Subsystem Code
4	A8	Mfg Module Code
5	A1	Program Language
6	F6.2	Timing - Mfg Est
7	I5	Instr - Mfg Est
8	I5	Data - Mfg Est
9	F5.1	CPS - Current
10	F6.2	Timing - Current
11	I5	Instr - Current
12	I5	Data - Current
13	I2	Year of Update
14	A2	Computer System
15	A30	Comments Text
16	A1	User Flag 1
17	A1	User Flag 2
18	A1	User Flag 3

In addition, since fields eleven through fifteen were not specified, SIMSIZ would enter default values of a blank for Alpha (A) formats and zeroes for Integer (I) and Fixed Point (F) formats. The important point to remember is

that once the field entry order has been specified, all records entered from the terminal must contain all of the specified fields in the specified order.

If one or more of the records being added does not have data for all of the specified fields, a value must still be entered for that field. If it is a numeric field enter "0", in lieu of a specific value. If it is an alpha field enter " " (a set of quote marks with a blank between them) to null out the field.

Figure 1 is an example of how the records for the C-141 simulator were entered through the Add function. Note that the first line contains the order of the fields being entered.

```
1,2,4,5,14,11,13,15
A,CN,' ',A,W1,1897,75,"RELOCATING AND ABSOLUTE LOADERS"
A,CR,' ',A,W1,11775,75,"COM STOR, EXEC, AND COM SUBROUT"
A,EP,CD2,A,W1,282,75,"PWR AVAIL FOR CKT BREAKERS"
A,EG,DA5,A,W1,212,75,"ENG RPM LOOP"
  .
  .
  .
```

Fig. 1. Sample Data Records

Test vs Real Add Function

The "test" mode of the Add function has been included to verify the sequence and the number of data fields on permfiles prior to adding the records to the data base. The "test" mode reads unformatted records, rearranges the fields into the order required by the data base, and

tests for the proper number of fields. If syntax or format errors are encountered in one of the records, or if a field was accidently excluded, the run will abort. The user may then correct the error on the permfile and re-initiate the test mode of the Add function. This cycle would continue until all errors on the permfile are corrected. The user then runs the Add function in the "Real" mode and adds the new records to the data base without the risk of entering faulty data.

SECTION IV

SELECTING/LISTING RECORDS FROM THE DATA BASE

Selecting Records from the Data Base

Records, when selected from the data base by the Select function, are marked (with an 'S' in the status flag) so that the same set of records can later be retrieved. There are two ways to select records from the data base and place them on the TEMP file: (1) do a New (N) select against the data base by specifying certain "select criteria," or (2) do a Previous (P) select against the data base for a set of records already marked with an 'S' flag. When an 'N' select run is made, records which do not meet the new selection criteria are marked with a 'G' in the status flag thereby de-selecting all records marked by a previous select run. The 'P' select option, which allows for re-selection of a previously selected set of records, was included for two reasons. First, checking the status flag field for an 'S' is faster than checking multiple fields against possibly complex selection criteria. Secondly, if the CREATE system should go down in the middle of your analysis you can quickly re-select your TEMP file contents using this option.

When the user specifies his selection criteria he is telling SIMSIZ what values, or range of values, he desires in the records to be selected. The values are specified by record field number. The fields are referred to as F1, F2, F3, . . . F18. If the user desires to select all the records on the data base with a value of 'A' in field one ('A' is the code for the C-141 simulator), he would enter the following string:

= F1 EQ 'A'#

(The "#" sign denotes the end of the criteria string.)

SIMSIZ would then check field one of every record on the data base and any records having an 'A' in that field would be marked, selected, and written to the TEMP file.

If the user wanted all records with 'A' or 'J' in field one he would enter the following string:

= F1 EQ 'A' OR 'J'#

If he wanted 'A' or 'J' or 'M' he enters:

= F1 EQ 'A' OR 'J' OR 'M'#

The "OR" is referred to as a "Minor Function Link" since it is not set off by commas. The "OR" function is the only minor function link SIMSIZ recognizes.

If the user wanted to select all records with field one equal to 'A' and field two (the function code) equal to Hydraulics (HD) he would enter:

= F1 EQ 'A', AND, F2 EQ 'HD'#

The "AND" is referred to as a "Major Function Link" because both F1 EQ 'A' and F2 EQ 'HD' must be true in order for a record to meet the select criteria. Note that the major function link is set off by commas. Both OR and AND are acceptable major function links. The 'A' and 'HD' are referred to as alphanumeric literals and are delimited by quotes. Fields one through five, and fourteen through eighteen are alphanumeric.

When selecting records based on numeric values, as in fields six through thirteen, numeric literals are delimited by "/". For example:

= (F11 GE /250/ ,AND, F10 LE /5.05/) ,OR, F13 EQ /76/#

This string will select all records with a Current Instruction Size (F11) greater than or equal to 250, AND, with Current Timing (F10) less than or equal to 5.05 ,OR, alternatively, with Year of Update (F13) exactly equal to seventy-six.

The permissible values for comparison functions are the same as appear in ANSI standard FORTRAN code: EQ (equal to), NE (not equal to), LT (less than), GT (greater than), LE (less than or equal to), and GE (greater than or equal to).

Parentheses may be used to prioritize criteria within the string. However, it is recommended that the use of parentheses be kept to a minimum.

If a criteria string cannot be completed on one line, it may be continued on the following line. SIMSIZ will continue to request information by prompting the user with the "=" sign until it encounters the "#" delimiter. Some examples of criteria strings follow:

= F1 NE 'A'#

- select if field one is any value other than 'A'.

= F1 GT 'E'#

- select if field one has an octal value greater than 'E' (i.e., select if F1 equals 'J' or 'M').

= F12 LT F8#

- select if field twelve (Current Data size) is less than field eight (Manufacturer's Estimated Data size).

Listing Records

From the Data Base

There are two output formats available for listing data records. The format is selected by the user during the SIMSIZ Log-On procedure. The 132-column option, selected by entering a number greater than or equal to 120, formats the data in single-line output. The 80-column option, selected by entering a number less than 120, formats the data into a double-line output.

Selecting List (L) at the "FUNCTION?" level allows the user to list data from either the data base, or from the TEMP file. Within the List function, the 'A' option will

list all the records on the data base. The records will be sequentially numbered according to their relative order in the data base. A listing may be terminated at any time by pushing the "break" key.

From the SIMSIZ TEMP File

The 'T' option will list the records on the TEMP file. They will be sequentially numbered in the order in which they appear on the TEMP file.

The 'R' option allows the selective listing of a range of records from the TEMP file. The user must specify the number of the first and the last records to be listed. The 'R' option is also available under the Modify function. This listing capability is limited to the Range mode of operation.

A user's sequential permfile can be listed using SIMSIZ. However, it must first be written to the TEMP file via the GET function. Note: only formatted data records may be listed by SIMSIZ. The GET function reads the formatted permfile and writes to the TEMP file.

SECTION V

MODIFYING/UPDATING THE DATA BASE

Data Records

Marking, Changing, or Deleting Selected Records

A data base record, to be changed or deleted, must first be selected and placed on the SIMSIZ TEMP file. The record can then be changed, user-marked, or delete-marked using the SIMSIZ Modify (M) function. Once the records on the TEMP file are changed to the user's satisfaction, the SIMSIZ Update (U) function is used to overwrite the original record on the data base with the modified record. When a record on the TEMP file is modified, its status flag is set to 'M'. If a record is delete-marked, its status flag is set to 'D'.

Updating the Data Base with Selected Records

The Update function scans each record on the TEMP file. If a record's status flag is set to 'M', a search is made for that record's original location in the data base. When the original record is located, it is over-written with the modified record. If the record on the TEMP file has been delete-marked, the status flag in the original record is set to 'D' for delete. The delete count for the proper

Function Code in the directory is also updated. Once a data-base record has been delete-marked it cannot be selected by the Select function. Records on the TEMP file with status flags other than 'M' or 'D' are ignored by the Modify function.

Directory Section

Adding, Changing, or Deleting Entries

The SIMSIZ Directory Modify Function (D) is used to add, change, or delete entries, by function code, in the directory section of the data base. The SIMSIZ Directory Section appears in Figure 2. If a directory entry (for example HD) is deleted, all of the data records with the function code (HD) are deleted.

New function codes can be created; the maximum is a total of forty codes. Function codes are unique two-character codes. The twelve-character function names, contained in the directory, can be changed using the 'D' function.

SYSTEM NAME	SYST CODE	DATA STARTING SECTOR	DATA ENDING SECTOR	SECTORS WITH DATA	NUMBER DELETED RECORDS
AURAL CUES	AC	9	234	2	0
AUX/EXT POWR	AP	10	258	2	0
ARMA/STORES	AS	11	212	5	0
AVIONICS	AV	12	201	8	0
COCPIT DISPL	CD	13	251	14	0
COMMUNICATON	CM	14	267	7	0
COMP,NON-REA	CN	15	250	6	0
COMP,REAL TI	CR	16	335	95	0
DATA FILES	DF	17	257	23	0
ENGINES(S)	EG	18	204	9	0
ELEC POWER	EP	19	236	8	0
ENVIRONMENTL	EV	20	231	6	0
ELEC WARFARE	EW	21	271	7	0
FLT CONT SUR	FC	22	259	11	0
FUEL SYSTEMS	FS	23	232	5	0
GROWTH PROVI	GP	24	24	1	0
HYDRAULICS	HD	25	229	5	0
INSTRUCTIONL	IN	26	272	27	0
LANDING GEAR	LG	27	163	4	0
MOTION	MO	28	242	5	0
NAVIGATION	NV	29	268	32	0
RADAR	RD	30	223	17	0
SIM ENVIRONT	SE	31	230	21	0
SPECL SYSTMS	SS	32	269	8	0
TACTICS	TC	33	260	3	0
VIS,REAL TIM	VR	34	261	4	0
VIS, SUPPORT	VS	35	35	1	0

Fig. 2. SIMSIZ Directory Section

Information Section

CPU and Language Information

The user selects the 'H' function to add, change or delete entries in the header information section of the data base. The 'L' option of the 'H' function can be used to list header subsections such as simulator type, type of CPU used, language, and explanatory text. The 'U' option of the 'H' function is used to change, add, or delete information in the header subsections.

There is space for up to fifteen entries in the CPU subsection. Each data record in the data base contains a two-character computer code which corresponds to one of the codes in the CPU information subsection. This information is used to standardize data units used by the Run function during a sizing run. Figure 3 contains the CPU and Language information sections.

The Bytes/Word value in a CPU information section is the conversion factor used to adjust the data in fields seven, eight, eleven, and twelve of a data record. Bytes are used as the standard for the Sizing function. For example, if a contractor reports his sizing data on a software module in computer words, the words are converted to bytes using the CPU Byte Conversion Factor. If a contractor reports in bytes, the conversion factor is 1.0. A byte is eight bits long. Hence a computer using 32-bit words has a byte/word conversion factor of 4.0.

The CPU Timing Conversion Factor is an additional value to be used to adjust a module's timing data (fields six and ten) to compensate for differences in execution speed of CPUs. These factors can be adjusted by the user.

The intent of the Language Conversion Factor is to compensate for differences in speed and size requirements of modules that are written in different computer languages. These factors can be altered by the user as necessary. Up to ten computer languages can be included in the language subsection.

--------CPU INFO---------				----LANGUAGE INFO---		
CPU CODE	CPU NAME	BYTES/ WORD	TIME CONV	LANG CODE	LANGUAGE NAME	CONV FACTOR
W1	SEL 840	3.0	1.00	F	FORTRAN	1.00
W2	SEL 840A	3.0	1.00	A	ASEMBLER	1.00
B1	INTERDAT	1.0	1.00	G	GMAP	1.00
W3	SEL32/35	4.0	1.00	P	PLACE	1.00
W4	HARRIS/5	3.0	1.00	M	MIXED	1.00
B2	UNITY	1.0	1.00	U	UNKNOWN	1.00
W5	SEL32/55	4.0	1.00	D	DATAONLY	1.00
M1	MULTI BY	5.0	1.00			

Fig. 3. CPU and Language Information Section

Simulator System Information

The simulator information subsection contains space for an entry for each simulator maintained in the data base. Figure 4 shows the information contained in the Simulator Information Subsection. A maximum of twenty entries can be created. When information on a new simulator

SIM CODE	SIMULATOR NAME	MISS CODE	CMPLXT FACTOR	INFO TEXT SECTOR NO	INSTRUCTIONS (IN BYTES)	DATA (IN BYTES)	TIMING (MSEC/SEC)
A	C-141		1.0	0	109,239	0	0.
B	C-5		1.0	0	163,740	45,003	0.
C	C-130		1.0	0	1,074,306	456,961	4,867.68
D	C-141CPT		1.0	0	122,672	53,880	531.00
E	C-5CPT		1.0	0	175,281	0	571.69
J	F-16		1.0	0	437,960	230,616	2,140.66
M	A-10		1.0	0	836,872	217,280	1,753.55

Fig. 4. Simulator Information Subsection

program is incorporated into the data base, a new entry must be created in this subsection.

The complexity factor allows the users to automatically apply an adjustment to all sizing runs against data from one or more simulators. All sizes and times in a data record are multiplied by the complexity factor during a sizing run.

Each entry in the simulator information subsection has a corresponding entry in the Text Information subsection. Space has been allocated to include text which explains unique characteristics of each simulator program. Information is entered into the Text section one line at a time (up to eighty characters per line). Text is saved by storing five lines per sector. The user can enter as much text as he desires but should bear in mind that each sector used for text is one less sector available for storing data records.

SECTION VI

SAVING/RETRIEVING DATA RECORDS ON PERMFILES

How to Use the SIMSIZ Put Function

The SIMSIZ Put (P) function takes formatted data records from the TEMP file and writes them to a user created sequential permfile. This permfile must have been created prior to logging-on to SIMSIZ. The permfile must be large enough to hold all the records which the user plans to store on the file. It is suggested to create the permfile with a small (1 LLINK) current size, but allow for growth by specifying a large (30 LLINK) maximum allowable size (see Section II). The file can then grow, as the Put Function writes to the permfile, to the size needed to hold all of the data. If 30 LLINKS is not large enough, the file may be modified or re-created with a larger maximum allowable size.

There are two options available within the Put Function: (1) write all (A) records on the TEMP file to the permfile, and (2) write only the records which have been modified (M) to the permfile.

How to Use the SIMSIZ Get Function

The Get function (G) reads formatted data records from user-supplied sequential permfiles and writes them onto the SIMSIZ TEMP file. The records must have been previously

written to the permfile by the Put function. No flags are reset. All data is transferred as ordered onto the permfile.

SECTION VII

USING THE SIMSIZ SIZING FUNCTION

Option 1 - A Sizing Run Using the Entire Data Base

Option 1 performs a sizing run against the entire SIMSIZ data base. The sizing is reported by Function Code (Func code) and by Simulator Code (Sim code). If the simulator package used in sizing the run has no data for a given Func code then no output is printed for that combination of Sim code and Func code.

The values printed for each SIM code are calculated by the following equations:

Note: F1 = Field 1 of the selected data record, F9 equals Field 9, etc.

LCF = Language Conversion Factor

SCF = Simulator Complexity Factor

CBF = CPU Byte Conversion Factor

CTF = CPU Timing Conversion Factor

Maximum Iteration Rate = The maximum value of field 9 from among the selected records.

Current Total Time = Sum of F9 X F10 X LCF X SCF (over all selected records).

Average Time per Cycle = Current Total Time/Maximum Iteration Rate.

Current Total Instruction Size = Sum of F11 X LCF X SCF X CBF (over all selected records).

Current Total Data Size = Sum of F12 X LCF X SCF X CBF (over all selected records).

Manufacturer's Est. Timing = Sum of F6 X F10 X LCF X SCF (over all selected records).

Manufacturer's Est. Instruction Size = Sum of F7 X LCF X SCF X CBF (over all selected records).

Manufacturer's Est. Data Size = Sum of F8 X LCF X SCF X CBF (over all selected records).

Grand totals in all categories, except Max Iteration Rate and Average Time Per Cycle, are reported by Sim code. The grand total values of Current Timing, Instruction Size, and Data Size are written into the simulator information subsection of the data base each time this option runs to normal termination. A listing of the information section will provide totals for all simulator systems in the data base.

Option 2 - A Sizing Run Using the SIMSIZ TEMP File

The principle difference between Option 1 and Option 2 is that Option 2 makes its sizing run against records contained on the TEMP file. Data are grouped and reported

in the same manner. Output values are computed using the same record fields and conversion factors as under Option 1. The Grand Totals are printed but are not stored in the information subsection of the data base.

Option 3 - A Sizing Run Using Selected Data Base Records

Complexity Factor Option

This option provides the flexibility of compensating for complexity factors during a sizing run. Separate complexity factors can be incorporated as the output for each Function Code is presented. The selected data record values are adjusted for complexity and written to the TEMP file. Records can then be modified, if necessary, and saved on a permfile or added to the data base as a new simulator package. If no complexity adjustment is desired a complexity factor of 1.0 should be entered.

Weighting Factor Option

This option provides the capability to weight the size and timing data by each Sim code at each functional category. For example, suppose the user is interested in estimating the average size of the Hydraulics function (HD) for a proposed simulator package. From his analysis of the hydraulic system complexity and specifications, he estimates that the size should be somewhere between the size of the 'HD' modules in the F-16 and the 'HD' modules in the A-10

simulator. Let's say that he elects to weight the results in favor of the F-16 (60/40). He would specify the number 60 for the F-16 data and the number 40 for the A-10 when prompted by the program. The weighted average for the estimated size of the proposed Hydraulics function is the sum of 60 percent of the F-16 values, plus 40 percent of the A-10 values. When the weighting option is not used a "simple average" of the data values is calculated.

New Record Option

This option allows the user to create new formatted data records from either the simple or weighted average generated by Option 3. If the user wishes to create new records during a run, he supplies SIMSIZ with a single alphanumeric character to be used as the Sim code for the records that he creates during the run. As the averages for each functional category are printed, SIMSIZ gives the user the option of creating a record after each group of averages is calculated. A formatted data record is generated placing appropriate values in the fields listed in Table B. The new records are written to the TEMP file. They can then be added to the data base by the Add function, placed on the user's permfile with the Put function, or retained on the TEMP file to be further modified by the Modify function.

TABLE B

FORMATTED DATA-RECORD FIELD VALUES

Field	Description
F1	Single character specified by the user
F2	Function Code under which these average values were generated
F3	Null
F4	Null
F5	F for FORTRAN - the sizing standard
F6	Manufacturer's Est. for Timing
F7	Manufacturer's Est. for Instruction Size
F8	Manufacturer's Est. for Data Size
F9	Maximum Iteration Rate
F10	Current Total Timing
F11	Current Total Instruction Size
F12	Current Total Data Size
F13	Current Year
F14	Null
F15	Null
F16	0
F17	0
F18	0

SECTION VIII

UTILITY ROUTINES FOR OBTAINING LISTINGS

This section was included to acquaint the user with two of the utility routines, available to CREATE users, which will be useful in data-base management. These Utility routines are small batch jobs which do not run under SIMSIZ. They may only be run from the 'star' (*) level (see Section I). The routines will generate a complete listing (dump) of all records on either a data-base file, or a user permfile of formatted records. The output will be available for pickup in the CREATE Production Control (I/O) shop in the basement of building 262, near Post D.

In order to run jobs in the Batch mode on CREATE the user must first log-on to the CREATE System as in Section I. The log-on is essentially the same, except that the user must answer "CARDIN" instead of "FORT N" to the "SYSTEM?" question. The user may then run batch jobs directly from the terminal.

After the user responds with "CARDIN" the system will ask if he wants to attach an old or a new file for the run. The response here depends upon which of the following two programs the user intends to exercise.

Listing an Entire Data Base

Enter:

```
OLD OR NEW-OLD BASEDMP
ready
*  RUN
```

If the user wishes to dump the contents of a different data-base file then certain alterations must be made to the BASEDMP program. Line number 1690 must be changed. Enter:

```
LIST 1690
```

The system will respond with

```
1690$:PRMFL:10,R,R,SIMCOM/SIMDATA
1700$:ENDJOB
ready
*
```

The user must now re-enter line 1690 changing only the CAT/FILE portion as appropriate. For example:

```
1690$:PRMFL:10,R,R,Userid/Filename
```

The system will respond with an '*'. The user may then enter 'RUN' and the system will respond with a SNUMB number. Record the SNUMB number to identify your printout.

Listing Formatted Permfiles

Enter:

```
OLD OR NEW-OLD PERMDMP
ready
*
```

List line 1470, as above, and change only the CAT/FILE portion as appropriate. Run the job and record the SNUMB.

APPENDIX B

DEFINITIONS OF STANDARDIZED MODULE CATEGORIES

Armament/Stores (AS)--modules that simulate the aerodynamic effects of carrying/releasing external munitions, fuel tanks, RPVs, or ECM pods on the aircraft.

Aural Cues (AC)--modules that provide sound simulations of all phases of ground and flight operations. Noise from engines, APUs, hydraulic pumps, air turbine motors, and weapons release are some examples.

Auxiliary/External Power (AP)--modules that simulate electrical and bleed air requirements prior to engine(s) start and after engine(s) shutdown.

Avionics (AV)--modules that simulate aircraft performance instrumentation. Performance indicators include airspeed, altitude, vertical velocity, attitude, heading, and course. Central air data computers, if included, are considered avionics.

Cockpit Displays (CD)--modules that simulate visual cockpit warning devices such as master caution systems, fire warning systems, and fire suppression systems. Special visual devices such as Heads-Up Display (HUD) are categorized as cockpit displays.

Communication (CM)--modules that simulate the aircraft's communication systems (FM, HF, VHF, and UHF radios).

Transponders (IFFs) are categorized as communication.

Computational Non-real Time (CN)--modules that fulfill support and diagnostic functions such as compilers, loaders, maintenance and test routines.

Computational Real Time (CR)--modules that perform real time computer operations such as executive routines, handlers, and common subroutine libraries.

Data Files (DF)--modules that contain only data entries. These modules contain no executible instructions and therefore no processor time requirements. Data file modules are functionally coded into standardized module categories. The codes are contained in the data-record field for subsystem code.

Electrical Power (EP)--modules that simulate electrical system performance, electrical power indicators, and aircraft lighting.

Electronic Warfare (EW)--modules that simulate detection masking, signal jamming, threat assessment, and other electronic warfare functions.

Engines (EG)--modules that simulate aircraft engine performance and instrumentation. This category includes high and low speed turbine RPMs, EGT, oil

quantity and temperature, engine stall characteristics, and anti-ice systems.

Environmental (EV)--modules that simulate the pneumatic system. This category includes the bleed air system performance (during flight), oxygen system useage, and the cabin temperature/pressurization indicators.

Flight Controls (FC)--modules that simulate flight-control aerodynamics. This category generally includes flight-control surfaces, autopilot, stability augmentation, and trim systems.

Fuel System (FS)--modules that simulate fuel related activities. Refueling (aerial if included), defueling, aircraft fuel useage, and fuel indicators are included in this category.

Growth Provisions (GP)--modules added to the original design specifications that result from user initiated ECPs. Examples include the DRLMs for the F-16 and the INS for the C-5 CPT.

Hydraulics (HD)--modules that simulate hydraulic system performance and indications. These modules perform the functions of values, pumps, and pressure regulators. Hydraulic system indicators and warning lights are also included.

Instructional (IN)--modules that provide the simulator instructor with the ability to control the simulation environment. Instructor training aids such as video/audio recorders, CRTs for visual displays of simulator position, and weather/malfunction control devices are categorized as instructional.

Landing Gear (LG)--modules that simulate the operation of the aircraft's landing gear, brakes, nose-wheel steering, anti-skid system, and associated indicators.

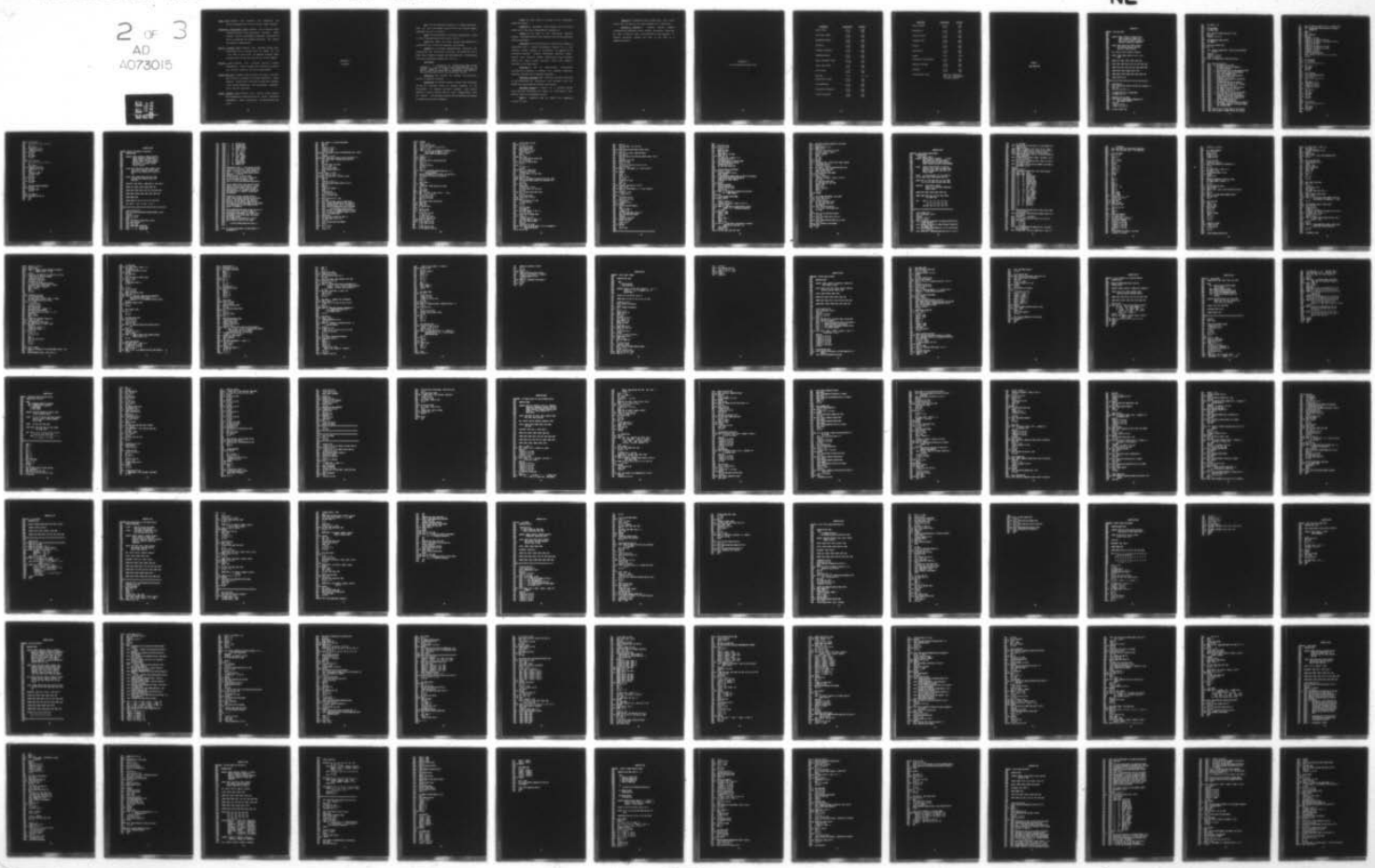
Motion (MO)--modules that compute required inputs to the mechanical motion system of the simulator. These include simulated motion during taxi, takeoff, flight (in smooth and in rough air), and landing. Mechanoreceptors and ejection, if applicable, are categorized as motion.

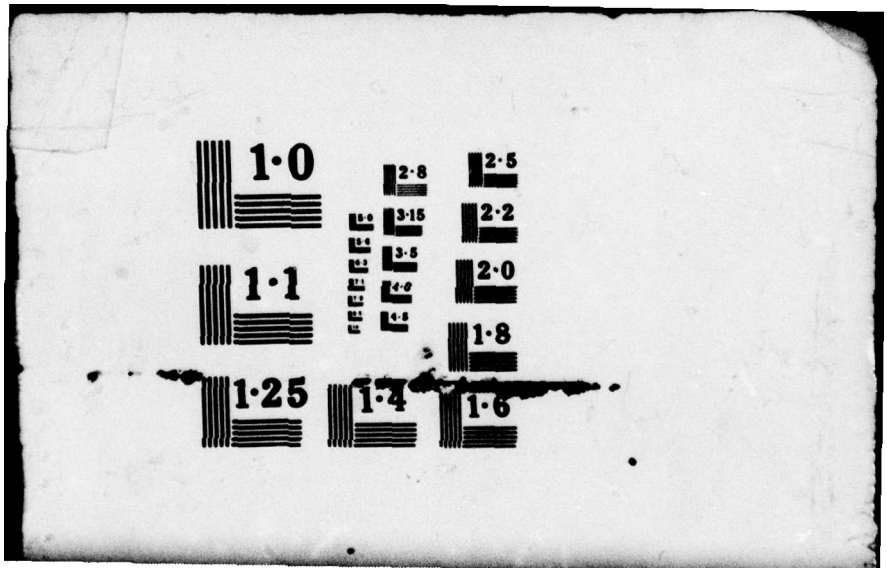
Navigation (NV)--modules that simulate inputs necessary to perform aircraft navigation. These include computations for magnetic variation, navigation aid station location, aircraft position, position freeze, and celestial positions. Navigation aids such as TACAN, VOR, ILS, OMEGA, area nav, and INS are categorized as navigation.

AD-A073 015 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL--ETC F/G 9/2
A PARAMETRIC MANAGEMENT TOOL FOR ESTIMATING SIMULATOR SOFTWARE --ETC(U)
JUN 79 G N FREY, K L WILDUNG

UNCLASSIFIED AFIT-LSSR-4-79A NL

2 OF 3
AD
A073015




1·0
1·1
1·25
2·8
3·15
3·5
4·0
1·4
2·5
2·2
2·0
1·8
1·6

Radar (RD)--modules that simulate the operation and visual presentations of the aircraft radar systems.

Simulation Environment (SE)--modules that simulate flight characteristics of the particular aircraft. These modules provide aerodynamic equations, equations of motion, equations for ground reactions, and weight and balance computations.

Special Systems (SS)--modules that simulate unique characteristics of an aircraft such as MADAR (on the C-5), SKE (on the C-130), or special aircraft model features such as the HC-130P and the C-130 (AWADS).

Tactics (TC)--modules that simulate tactical fighter engagements. These include such features as targeting, firing, tracking, and kill computations.

Visual Real (VR)--modules that simulate the visual environment during all phases of aircraft operation. These modules include such functions as position computations, image generation, wind and weather compensation, and CRT interface.

Visual Support (VS)--modules that fulfill visual support and diagnostics functions such as visual data-base management, scene generation, and maintenance and test.

APPENDIX C

GLOSSARY

Bit--"is the smallest element in a binary character. Each bit has two states, either off or on, thusthe representation of 0 or 1 [7:212]."

Byte--"is the smallest individual addressable group of bits--conventionally eight bits [12:12]."

Code--the means by which design requirements are translated into a form the computer can process.

CREATE--is an acronym (Computational Resources for Engineering And Simulation Training and Education) for a large scale computer system that provides both time-sharing and batch processing capability (10:2.2).

Data base--

> . . . a collection of interrelated data stored together . . . to serve one or more applications in an optimal fashion; a common and controlled approach is used in adding new data and in modifying and retrieving existing data within the data base [12:19].

Debugging--the process of finding and correcting errors in computer software.

Modularity--a software design concept that advocates the use of a building block or modular approach in the development of complex software systems. Each module performs a given function more or less independently from the others. The software system can be modified by adding or deleting particular modules.

Octal--an octal value is a number in the base-eight numbering system.

Permfile--a permanent mass storage disk file which exists until the user intentionally releases it.

SIMSIZ--is the name of the interactive computer program developed during this research for sizing simulator software packages.

Simulator--an Aircrew Training Device that ranges in complexity from a simple procedures trainer to a high fidelity system capable of simulating all aspects of an aircraft's mission. Cockpit Procedures Trainers (CPTs), Partial Task Trainers (PTTs), Operational Flight Trainers (OFTs), and Weapon System Trainers (WSTs) are commonly referred to as simulators.

Software--a set of step-by-step instructions required by a computer to perform its intended function, commonly referred to as computer programs.

Software engineer--ASD software engineers assigned to the Directorate of Engineering and matrixed into the SIMSPO for computational systems engineering.

Software module--a subunit of a software system which has been identified as a means of functionally segmenting code into manageable sizes.

TEMP--a tempfile used by SIMSIZ for temporary storage of data.

Tempfile--a temporary mass storage disk file which exists only as long as the using program is in execution.

Top-down design--a software design concept incorporating modularity and a smooth, continuous execution of code literally from top-to-bottom of the program; it reduced branching forward and back in the flow in a haphazard manner.

APPENDIX D

F-15 VALIDATION SIZING PLAN

FUNCTION	SIMULATOR	WEIGHT
Aural Cues	F-16	60%
	A-10	40%
Auxiliary Power	F-16	50%
	A-10	50%
Armament/Stores	F-16	80%
	A-10	20%
Avionics	F-16	20%
	A-10	80%
Cockpit Displays	F-16	60%
	A-10	40%
Communications	F-16	50%
	A-10	50%
Comp. Non-Real Time	A-10	70%
	C-130	30%
Comp. Real-Time	F-16	60%
	A-10	40%
Data Files	F-16	80%
	A-10	20%
Engines	F-16	(X 1.5)
Electrical Power	F-16	70%
	A-10	30%
Environmental	F-16	50%
	A-10	50%
Electronic Warfare	F-16	90%
	A-10	10%
Flight Controls	F-16	20%
	A-10	80%

FUNCTION	SIMULATOR	WEIGHT
Fuel Systems	F-16	70%
	A-10	30%
Hydraulics	F-16	50%
	A-10	50%
Instructional	F-16	60%
	A-10	40%
Landing Gear	F-16	40%
	A-10	60%
Motion	F-16	60%
	A-10	40%
Navigation	F-16	80%
	A-10	20%
Radar	F-16	100%
Simulator Environment	F-16	80%
	A-10	20%
Special Systems	A-10	60%
	C-141	40%
Tactics	F-16	70%
	A-10	30%
Visual-Real Time	The F-15 presently does not incorporate a visual system.	

APPENDIX E

SIMSIZ SOURCE CODE

PROGRAM MAIN

```
1000CHAIN   SIMSIZ MAIN DRIVER.
1010C
1020C
1030      CHARACTER  FNAME*6, DATES*8(3), CARD*450, ATCH*20,
1040&                TODAY*8, SYSCOD*2(40), SYSNAM*12(40),
1050&                ANS*3, CPUC*2(15), CPUN*8(15), LANCC*1(10),
1060&                LANCN*8(10), SIMCOD*1(20), SIMNAM*8(20)
1070C
1080      INTEGER  SECNO, LSIZE, DIR(3), INFO(3), DATAS(3),
1090&              CONT, DSSEC(40), DESEC(40), DNSEC(40),
1100&              NDEL(40), FC, RNUM, SIMTXT(20)
1110C
1120      REAL  CPUS(15), CPUT(15), LANCS(10), SIMFAC(20)
1130C
1140      LOGICAL  DIREUP, INFOUP, DATAUP, LA, LB, LC, LD, LE,
1150&              WINFO
1160C
1170      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
1180C
1190      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
1200C
1210      COMMON /INSEC/ CPUC, CPUN, CPUS, CPUT, LANCC, LANCN, LANCS
1220C
1230      COMMON /INFOS/ SIMCOD, SIMNAM, SIMFAC, SIMTXT
1240C
1250      COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DNSEC, NDEL
1260C
1270      COMMON /TERMTYPE/ LINE
1280C
1290*****************************************************************
1300C
1310* - - ATTACH MAIN FILE.
1320      PRINT, "ENTER CAT/FILE STRING OF DATA BASE FILE, FOLLOWED BY ';'."
1330      READ, ATCH
1340      CALL ATTACH (10, ATCH, 3, 1,,)
1350C
1360* - - SET RANDOM RECORD SIZE AT 76 WORDS/RECORD.
1370      CALL RANSIZ (10, 76, 1)
1380C
1390* - - DETERMINE OUTPUT LINE LENGTH.
1400 8    PRINT, "WHAT IS THE MAX NUMBER OF CHARACTERS/LINE ",
1410&            "YOUR TERMINAL CAN PRINT?"
1420      READ, LINE
1430      IF(LINE.LT.1) GO TO 8
1440      IF(LINE.LT.120) LINE = 80
1450C
1460* - - SET SIZE OF OUTPUT LINE.
```

```
1470      CALL FPARAM (1, 132)
1480C
1490* - - SET BREAK ADDRESS.
1500      CALL BRKADR ($200)
1510C
1520      PRINT, "HAS THIS FILE BEEN INITIALIZED (Y OR N)?"
1530      READ, ANS
1540      IF(ANS.EQ.'N') GO TO 10
1550C
1560* - - LOAD COMMONS IDSEC, DRSEC, AND MISC.
1570      CALL RWHEAD (1)
1580C
1590* - - SET NSEL TO A NEGATIVE VALUE.
1600 10   NSEL = -10
1610C
1620      PRINT, "WELCOME TO PROGRAM SIMSIZ.  WOULD YOU LIKE INSTRUCTIONS ",
1630&           "(Y OR N)?"
1640      READ, ANS
1650      IF(ANS.EQ.'Y') GO TO 15
1660      WINFO = .F.; GO TO 25
1670 15   PRINT, "THIS PROGRAM AND ITS ASSOCIATED DATA BASE....."
1680C
1690C
1700      PRINT, "SIMSIZ CAN BE USED TO:"
1710      PRINT, "   A - ADD RECORDS TO THE DATA BASE."
1720      PRINT, "   D - ADD, CHANGE, DELETE, OR LIST RECORDS IN "
1730      PRINT, "       THE DATA BASE DIRECTORY."
1740      PRINT, "   B - BUILD DATA BASE FILE WITH ID AND DIRECTORY SECTIONS."
1750      PRINT, "   M - MODIFY (OR DELETE MARK) ONE OR MORE RECORDS"
1760      PRINT, "       ON THE 'TEMP' (TEMPORARY) FILE."
1770      PRINT, "   U - UPDATE THE DATA BASE WITH MODIFIED/DELETED"
1780      PRINT, "       RECORDS FROM THE 'TEMP' FILE."
1790      PRINT, "   P - PUT MODIFIED 'TEMP' RECORDS ON A USER-SUPPLIED"
1800      PRINT, "       SEQUENTIAL PERM FILE."
1810      PRINT, "   G - GET RECORDS FROM USER-SUPPLIED SEQUENTIAL"
1820      PRINT, "       PERM FILE (BUILT PREVIOUSLY BY 'PUT' FUNCTION)"
1830      PRINT, "       AND PLACE RECORDS ON 'TEMP' FILE."
1840      PRINT, "   H - ADD, CHANGE, DELETE OR LIST DATA IN THE DATA "
1850      PRINT, "       BASE 'HEADER' (ID AND INFO) SECTIONS."
1860      PRINT, "   L - LIST RECORDS ON 'TEMP' FILE."
1870      PRINT, "   R - RUN A SIZING ANALYSIS ON SELECTED DATA RECORDS."
1880      PRINT, "   S - SELECT MODE - USED TO SELECT ONE OR MORE "
1890      PRINT, "       RECORDS FROM THE DATA BASE (FOR MODIFICATION, "
1900      PRINT, "       DELETION, LISTING, OR USE IN SIZING ANALYSIS)"
1910      PRINT, "       AND PLACE THEM ON THE 'TEMP' FILE."
1920      PRINT, "   E - END - GO TO NORMAL TERMINATION."
1930      PRINT, "   I - INSTRUCTIONS - TO HAVE THE INSTRUCTIONS REPRINTED."
1940      PRINT, "   X - END - ABNORMAL TERMINATION.  DO NOT WRITE COMMONS"
1950      PRINT, "       BACK TO THE DATA BASE."
1960      PRINT, " "
1970      PRINT, "NOW ENTER THE LETTER WHICH IDENTIFIES THE OPTION YOU"
1980      PRINT, "WISH TO EXERCISE.  REMEMBER YOU MUST FIRST ESTABLISH"
```

```
1990      PRINT, "THE TEMP FILE (USING 'S' OR 'C') IF YOU WISH TO DO"
2000      PRINT, "ANYTHING OTHER THAN ADD RECORDS OR LIST HEADER ",
2010&           "INFORMATION."
2020      PRINT, " "
2030      WINFO = .T.
2040 25   PRINT, "FUNCTION?"
2050      READ, ANS
2060      IF(ANS.EQ.'A') GO TO 50
2070      IF(ANS.EQ.'U' .OR. ANS.EQ.'H') GO TO 90
2080      IF(ANS.EQ.'D' .OR. ANS.EQ.'L' .OR. ANS.EQ.'M') GO TO 60
2090      IF(ANS.EQ.'E') GO TO 900
2100      IF(ANS.EQ.'I') GO TO 15
2110      IF(ANS.EQ.'B' .OR. ANS.EQ.'R') GO TO 100
2120      IF(ANS.EQ.'P' .OR. ANS.EQ.'G') GO TO 100
2130      IF(ANS.EQ.'S') GO TO 80
2140      IF(ANS.EQ.'X') GO TO 910
2150      PRINT, "YOU MUST ENTER EITHER:"
2160      PRINT, "A, B, D, E, G, H, I, L, M, P, R, S, U, OR X."
2170      GO TO 25
2180C
2190* - - CALL ADD ROUTINE.
2200 50   IF (LA) GO TO 55
2210      LB = .F.; LC=.F.; LD=.F.; LE = .F.
2220      LA = .T.
2230 55   CALL LLINK("A    ")
2240      CALL ADREC
2250      GO TO 25
2260C
2270* - - DO LIST FUNCTION.
2280*     SEE IF LINK IS ALREADY IN CORE.
2290 60   IF(LC) GO TO 63
2300      LA=.F.; LB=.F.; LD=.F.; LE = .F.
2310      LC=.T.
2320      CALL LLINK ("C    ")
2330 63   IF(ANS.EQ.'M') GO TO 70
2340      IF(ANS.EQ.'L') GO TO 64
2350      IF(ANS.EQ.'D') GO TO 66
2360 64   FC = 0
2370      CALL LISTR (FC)
2380      GO TO 25
2390 66   CALL DIRMOD
2400      GO TO 25
2410 70   CALL TMOD
2420      GO TO 25
2430C
2440 80   IF(LB) GO TO 85
2450      LA=.F.; LC=.F.; LD=.F.; LE = .F.
2460      LB=.T.
2470      CALL LLINK("B    ")
2480 85   CALL SELREC
2490      GO TO 25
2500C
```

```
2510 90   IF(LD) GO TO 92
2520      LA = .F.; LB = .F.; LC = .F.; LE = .F.
2530      LD = .T.
2540      CALL LLINK ("D    ")
2550 92   IF(ANS.EQ.'U') GO TO 94
2560      IF(ANS.EQ.'H') GO TO 96
2570 94   CALL UPDATE
2580      GO TO 25
2590 96   CALL HEADER
2600      GO TO 25
2610C
2620 100  IF(LE) GO TO 110
2630      LA = .F.; LB = .F.; LC = .F.; LD = .F.
2640      LE = .T.
2650      CALL LLINK ("E    ")
2660 110  IF(ANS.EQ.'R') GO TO 120
2670      IF(ANS.EQ.'B') GO TO 130
2680      MODE = 1
2690      IF(ANS.EQ.'C') MODE = 2
2700      CALL PUTGET (MODE)
2710      GO TO 25
2720 120  CALL RUNSIZ
2730      GO TO 25
2740 130  CALL SIMBLD
2750      GO TO 25
2760C
2770* - - RESET BREAK ADDRESS AND CONTINUE.
2780 200  CONTINUE
2790      CALL BRKADR ($200)
2800      GO TO 25
2810C
2820 900  CALL RWHEAD (2)
2830 910  CALL DETACH (20, ISTAT, 0)
2840      STOP
2850      END
```

SUBROUTINE ADREC

```
C     ADREC  ROUTINE TO ADD RECORDS TO THE DATA BASE.
      SUBROUTINE ADREC
C
      CHARACTER  FNAME*6, DATES*8(3), CARD*450, FREC*90(5),
     &           TODAY*8, SYSCOD*2(40), SYSNAM*12(40), FC*1,
     &           FORM1*12, FORM2*61, CEE*1/'C'/, ATCH*20,
     &           NEWR*90(5), NSS*2/' '/, NEWB*450, ANSW*84,
     &           ANS*3, OSS*2/' '/, STR*8(15), LSTR*30,
     &           TBUF*450, TREC*90(5)
C
      INTEGER  SECNO, LSIZE, DIR(3), INFO(3), DATAS(3), SFLDS,
     &         CONT, DSSEC(40), DESEC(40), DNSEC(40), RNUM,
     &         NDEL(40), FL(15), REC(20), RP, FP, FLD(15), SP,
     &         TSEC
C
      LOGICAL  DIREUP, INFOUP, DATAUP, FIELD*1(15), DONE,
     &         RTN, RDATA, WINFO, MODSPLAY, SERIES, TEST,
     &         FORM, TEMP
C
      EQUIVALENCE  (CARD, FREC(1)), (NEWB, NEWR(1)), (TBUF, TREC(1))
C
      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
C
      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
C
      COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DNSEC, NDEL
C
      COMMON /MODBUF/ NEWB
C
      COMMON /ARRAYS/ RP, REC, FP, FLD, SP, STR, LSTR, NFLDS
C
      DATA  FORM1/"(T  ,A3,',')"/, OSS/'  '/, FC/' '/
C
C*****************************************************************************
C
      IF(.NOT. WINFO) GO TO 2
      PRINT, "DO YOU WANT INSTRUCTIONS FOR ADDING RECORDS (Y OR N)?"
      READ, ANS
      IF(ANS.EQ.'Y') GO TO 1
      WINFO = .F.; GO TO 2
    1 WINFO = .T.
      PRINT, "THE CONTENTS OF RECORD FIELDS 1 THRU 15"
      PRINT, "ARE AS FOLLOWS:"
      PRINT, "FIELD  FIELD"
      PRINT, "NUMBER FORMAT  DISCRIPTION"
      PRINT, "  1     A1    SIMULATOR CODE"
      PRINT, "  2     A2    FUNCTION CODE"
```

```
1470      PRINT, "  3     A2    SUBSYSTEM CODE"
1480      PRINT, "  4     A8    MFG MODULE CODE"
1490      PRINT, "  5     A1    PROGRAM LANGUAGE"
1500      PRINT, "  6     F6.2  TIMING - MFG EST"
1510      PRINT, "  7     I5    INSTR - MFG EST"
1520      PRINT, "  8     I5    DATA - MFG EST"
1530      PRINT, "  9     F5.1  CPS - CURRENT"
1540      PRINT, " 10     F6.2  TIMING - CURRENT"
1550      PRINT, " 11     I5    INSTR - CURRENT"
1560      PRINT, " 12     I5    DATA - CURRENT"
1570      PRINT, " 13     I2    YEAR OF UPDATE"
1580      PRINT, " 14     A2    COMPUTER SYSTEM"
1590      PRINT, " 15     A30   COMMENTS TEXT"
1600      PRINT, " "
1610      PRINT, "WHEN PROMPTED BY 'NEXT = ', ENTER THE DATA FOR FIELDS"
1620      PRINT, "1 THRU 15 OF THE NEXT RECORD.  SEPARATE THE FIELDS BY"
1630      PRINT, "COMMAS.  IF YOU WISH TO  SKIP A FIELD, THEN NULL IT"
1640      PRINT, "WITH ' '.  ALL FIELDS WITH EMBEDED BLANKS OR SPECIAL"
1650      PRINT, "CHARACTERS (; , $) MUST BE"
1660      PRINT, "ENCLOSED IN QUOTES.  IF YOU CAN'T ENTER"
1670      PRINT, "ALL THE DATA ON ONE LINE, JUST HIT 'RETURN' AND "
1680      PRINT, "YOU WILL BE PROMPTED WITH ANOTHER '=' SIGN"
1690      PRINT, "UNTIL ALL THE FIELDS HAVE BEEN ENTERED OR NULLED OUT."
1700      PRINT, " "
1710      PRINT, "YOU MAY, HOWEVER, PREFER TO BUILD THESE DATA STRINGS"
1720      PRINT, "AHEAD OF TIME AND PUT THEM ON A PERM-FILE.  YOU MAY"
1730      PRINT, "THEN DIRECT THIS ADD ROUTINE TO READ THE STRINGS"
1740      PRINT, "FROM THE PERM-FILE.  IF YOU CHOOSE THIS OPTION, THE "
1750      PRINT, "DATA STRINGS MUST STILL CONFORM TO THE ABOVE RULES"
1760      PRINT, "OF PUNCTUATION."
1770      PRINT, " "
1780      PRINT, "IN ADDITION, IF YOU WANT TO ADD DATA WHICH ARE "
1790      PRINT, "ACTUALLY MODIFIED RECORDS TAKEN FROM THE DATA BASE YOU"
1800      PRINT, "MAY DO SO.  THESE DATA MUST HAVE BEEN PLACED ON A "
1810      PRINT, "PERM-FILE BY THE 'PUT' ROUTINE.  SINCE THESE DATA"
1820      PRINT, "ARE ALREADY IN THE PROPER RECORD FORMAT, THEY ARE REFERRED"
1830      PRINT, "TO AS 'FORMATED' DATA, ALL OTHER INFORMATION (DATA"
1840      PRINT, "STRINGS) IS 'UNFORMATED'."
1850      PRINT, " "
1860      PRINT, "AFTER 5 RECORDS HAVE BEEN ENTERED (OR READ) THEY WILL ALL"
1870      PRINT, "BE DISPLAYED AT ONCE.  IF YOU WISH TO CORRECT A"
1880      PRINT, "RECORD ENTER THE RECORD NUMBER (1-5) FOLLOWED BY THE"
1890      PRINT, "FIELD NUMBER(S) (1-15) AND THE CORRECT FIELD VALUES. "
1900      PRINT, "WHEN YOU HAVE FINISHED "
1910      PRINT, "ALL 5 RECORDS WILL BE DISPLAYED AGAIN.  YOU WILL"
1920      PRINT, "THEN BE PROMPTED TO ENTER THE 'NEXT' RECORD."
1930      PRINT, " "
1940      PRINT, "     IF YOU ARE FINISHED ENTERING DATA, ENTER A Z."
1950      PRINT, " "
1960C
1970 2    PRINT, "IS THE DATA FOR THE RECORDS TO BE ADDED FORMATED 'F', ",
1980&           "OR UNFORMATED 'U'?"
```

```
1990      PRINT, "ENTER 'I' IF YOU WANT INSTRUCTIONS."
2000      READ, ANS
2010      FORM = .F.
2020      IF(ANS.EQ.'I') GO TO 1
2030      IF(ANS.EQ.'F') FORM = .T.
2040      IF(FORM) GO TO 2004
2050 2002 PRINT, "DO YOU WANT THE DATA TO BE ENTERED FROM A FILE (Y OR N)?"
2060      READ, ANS
2070      IF(ANS.NE.'Y') GO TO 3
2080 2004 PRINT, "ENTER CAT/FILE STRING OF DATA FILE FOLLOWED BY ';'."
2090      PRINT, "ENTER 'TEMP;' TO INDICATE SIMSIZ TEMP FILE."
2100      READ, ATCH
2110      TEMP = .F.
2120      IF(ATCH.NE.'TEMP;') GO TO 2006
2130      TEMP = .T.; FORM = .T.
2140      IF(NSEL.GT.0) GO TO 2008
2150      PRINT, "NO DATA ON TEMP FILE."
2160      RETURN
2170 2006 CALL DETACH (12, ISTAT,)
2180      CALL ATTACH (12, ATCH, 1, 0,,)
2190 2008 PRINT, "IS THIS A TEST 'T', OR REAL 'R' ADD?"
2200      READ, ANS
2210      TEST = .F.
2220      IF(ANS.EQ.'T') TEST = .T.
2230      PRINT, "DO YOU WANT RECORDS DISPLATED (Y OR N)?"
2240      READ, ANS
2250      NODSPLAY = .F.
2260      IF(ANS.EQ.'N') NODSPLAY = .T.
2270      RDATA = .T.
2280      IF(FORM) GO TO 35
2290      IF(.NOT. RDATA) GO TO 3
2300      READ (12, 40, END=4) ANSW
2310      GO TO 6
2320 4    PRINT, "NO DATA ON FILE ", ATCH
2330      GO TO 2002
2340 3    PRINT, "YOU MUST ENTER A SERIES OF NUMBERS WHICH"
2350      PRINT, "INDICATES THE ORDER YOU WILL USE IN ENTERING"
2360      PRINT, "THE RECORD FIELDS 1 THRU 15.  FOR EXAMPLE, IF"
2370      PRINT, "YOU WILL BE ENTERING DATA FIELDS IN THE ORDER"
2380      PRINT, "     1,2,3,7,8,9,4,5,6,11,12,10,13,14,15"
2390      PRINT, "THEN THAT IS THE NUMBER STRING YOU MUST ENTER."
2400      PRINT, "THE DATA WILL BE REORDERED INTERNALLY INTO THE"
2410      PRINT, "PROPER RECORD FORMAT."
2420 5    READ 40, ANSW
2430* - - LOAD VALUES INTO FLD ARRAY ONLY (MODE = 3).
2440 6    CALL ST2AR (ANSW, 3, 05, $7, $7)
2450      GO TO 10
2460 7    PRINT, "PLEASE RE-ENTER FIELD NUMBERS."
2470      GO TO 5
2480 10   DO 12 I = 1, 15
2490      FL(I) = FLD(I)
2500      FIELD(I) = .F.
```

```
2510 12   CONTINUE
2520      DO 20 I = 1, 15
2530      IF(FL(I).EQ.0) GO TO 20
2540      IF(FL(I).GE.1 .AND. FL(I).LE.15) GO TO 19
2550      PRINT 17, I, FL(I)
2560 17   FORMAT (1X, "THE ", I2, "TH VALUE IN THE STRING IS ", I2,
2570&           " AND IS OUTSIDE THE RANGE OF 1 TO 15.",/,
2580&           "PLEASE RE-ENTER THE COMPLETE STRING.")
2590      GO TO 5
2600 19   FIELD(FL(I)) = .T.
2610 20   CONTINUE
2620* - - CHECK TO SEE THAT ALL FIELDS ARE SPECIFIED.
2630      NFLDS = 15
2640      DO 30 I = 1, 15
2650      IF(FIELD(I)) GO TO 30
2660      PRINT 25, I
2670      RTN = .T.
2680 25   FORMAT (1X, "YOU DID NOT INDICATE WHERE FIELD ", I2,
2690&            " WOULD BE ENTERED.")
2700* - - ADD THIS FIELD NUMBER TO THE FLD ARRAY, BUT LEAVE THE STR
2710*     VALUE BLANK SO THE FIELD WILL BE NULLED OUT.
2720      FP = FP + 1
2730      FLD(FP) = I
2740      FL(FP) = I
2750      NFLDS = NFLDS - 1
2760 30   CONTINUE
2770      IF(RTN) PRINT, "DEFAULT VALUES WILL BE USED."
2780      RTN = .F.
2790      SFLDS = NFLDS
2800* - - RESET VALUES.
2810 35   DONE = .F.; SECNO = 0; CONT = 0; OSS = '  '; NSS='  '
2820      NSEC = 0; IPT = 0; ITOT = 0
2830      IF(.NOT. TEMP) GO TO 82
2840* - - READ FIRST TEMP SECTOR.
2850      TSEC = 1
2860      READ (20'TSEC, END=910, ERR=920) TBUF
2870 46   FORMAT (A84)
2880 58   FORMAT (A90)
2890* - - SET RECORD POINTER.
2900 82   I = 0
2910* - - READ NEXT RECORD.
2920 85   ITOT = ITOT + 1
2930      IF(.NOT. TEST) GO TO 851
2940      IR = MOD(ITOT, 10)
2950      IF(IR.NE.0) PRINT, "NEXT"
2960      IF(IR.EQ.0) PRINT 850, ITOT
2970 850  FORMAT (1X, "NEXT ", I3)
2980 851  IF(.NOT. RDATA) GO TO 86
2990      IF(.NOT. FORM) GO TO 855
3000      I = I + 1
3010      IF(.NOT. TEMP) GO TO 853
3020* - - GET NEXT RECORD OFF TEMP FILE.
```

```
3030      IF(ITOT.GT.NSEL) GO TO 854
3040 852  IPT = IPT + 1
3050      IF(IPT.GT.5) GO TO 8521
3060* - - CHECK FOR DELETE MARK.
3070      DECODE (TREC(IPT), 177) FG
3080      IF(FG.EQ.'D') GO TO 852
3090      NEWR(I) = TREC(IPT)
3100      GO TO 89
3110 8521 TSEC = TSEC + 1
3120      READ (20'TSEC, END=910, ERR=920) TBUF
3130      IPT = 0; GO TO 852
3140 853  READ (12, 50, END=854) NEWR(I)
3150      GO TO 89
3160 854  I = I - 1
3170      IF(I.LT.0) I = 0
3180      GO TO 868
3190 855  READ (12, 40, END=868) ANSW
3200      IF(ANSW.EQ.'#') GO TO 868
3210      CALL ST2AR (ANSW, 4, 12, $857, $868)
3220      GO TO 866
3230 857  PRINT 858, ANSW
3240 858  FORMAT (1X, "THE PROCESSING STOPPED AT THIS LINE:",/,A80,
3250&     "BUT THE ERROR PROBABLY OCCURED IN THE PRECEEDING LINE.")
3260      GO TO 990
3270 86   PRINT, "NEXT"
3280      READ 40, ANSW
3290      IF(ANSW.EQ.'#') GO TO 868
3300      CALL ST2AR (ANSW, 4, 05, $865, $865)
3310      GO TO 866
3320 865  PRINT, "PLEASE RE-ENTER THOSE FIELDS."
3330      GO TO 86
3340* - - CHECK FOR END OF DATA.
3350 866  IF(STR(1).NE.'#') GO TO 87
3360 868  DONE = .T.
3370* - - IF NO NEW RECORDS, SKIP TO END.
3380      IF(I.EQ.0) GO TO 210
3390      GO TO 92
3400 87   I = I + 1
3410* - - UPDATE THE RECORD.
3420      CALL RECMOD (I)
3430 89   CALL CONCAT (NEWR(I), 2, CEE, 1, 1)
3440      IF(I.LT.5) GO TO 85
3450* - - DISPLAY DATA IN STANDARD FORMAT.
3460 92   ITOTAL = 0
3470      IF(NODSPLAY) GO TO 150
3480      CALL DSPLAY (NEWB, I, ITOTAL, 1, 1)
3490      PRINT, "ARE THERE ANY ERRORS?"
3500      READ, ANS
3510      IF(ANS.EQ.'N') GO TO 150
3520 105  PRINT, "ENTER THE RECORD NUMBER (1 TO 5); FIELD NUMBER(S); ",
3530&           "AND NEW FIELD VALUES."
3540 107  READ 40, ANSW
```

```
3550      NFLDS = 0
3560      CALL STZAR (ANSW, 1, 05, $110, $110)
3570      GO TO 115
3580 110  PRINT, "PLEASE RE-ENTER COMPLETE CONTROL STRING."
3590      GO TO 107
3600 115  PRINT, "THIS IS THE WAY I READ YOUR INPUTS:"
3610      CALL DARAY (1)
3620      PRINT, "DO YOU WISH TO RE-ENTER YOUR CONTROL STRING (Y OR N)?"
3630      READ, ANS
3640      IF(ANS.EQ.'Y') GO TO 105
3650C - - CHECK FIELD VALUES.
3660      DO 125 J = 1, FP
3670      IF(FLD(J).GE.1 .AND. FLD(J).LE.15) GO TO 125
3680      PRINT 120, FLD(J)
3690 120  FORMAT (1X, "FIELD NUMBER", I3, " IS OUT OF LIMITS.")
3700      GO TO 110
3710 125  CONTINUE
3720      ITOTAL = 0
3730      SERIES = .F.
3740      DO 140 J = 1, RP
3750      NUM = REC(J)
3760      IF(NUM.GT.0) GO TO 130
3770      SERIES = .T.
3780      NUM = -NUM
3790 130  IF(NUM.GE.1 .AND. NUM.LE.I) GO TO 135
3800      PRINT 132, NUM
3810 132  FORMAT (1X, "RECORD NUMBER ", I2, " IS OUT OF BOUNDS.")
3820      GO TO 110
3830 135  IF(SERIES) GO TO 136
3840      K = NUM; GO TO 137
3850 136  K = K + 1
3860      IF(K.LE.NUM) GO TO 137
3870      SERIES = .F.; GO TO 140
3880 137  CALL RECMOD (K)
3890      IF(SERIES) GO TO 136
3900 140  CONTINUE
3910C - - DISPLAY THE MODIFIED RECORD(S).
3920      NUMB = 1
3930      IF(RP.GT.1) NUMB = I
3940      IF(NUMB.GT.1) K = 1
3950      CALL DSPLAY (NEWB, NUMB, ITOTAL, K, K)
3960      PRINT, "ANY ADDITIONAL CHANGES?"
3970      READ, ANS
3980      IF(ANS.EQ.'Y') GO TO 105
3990C - - REPLACE FIELD VALUES.
4000      DO 145 J = 1, 15
4010      FLD(J) = FL(J)
4020 145  CONTINUE
4030      FP = 15
4040      NFLDS = SFLDS
4050C
4060**************************************************************
```

```
4070C
4080* - - WRITE DATA TO FILE.
4090 150  IF(TEST) GO TO 205
4100      DO 200 IC = 1, I
4110* - - NUMBER THE RECORD.
4120      RNUM = RNUM + 1
4130      ENCODE (NEWR(IC), 151) RNUM
4140 151  FORMAT (T87, I4)
4150      CALL CONCAT (NSS, 1, NEWR(IC), 3, 2)
4160      IF(NSS.EQ.OSS) GO TO 175
4170* - - THIS RECORD BELONGS IN A DIFFERENT SECTOR.
4180*     WRITE THIS SECTOR BACK TO THE FILE.
4190      IF(SECNO.EQ.0) GO TO 152
4200      CALL WSEC (CARD, SECNO, CONT)
4210C
4220* - - LOOK FOR A MATCH IN THE DIRECTORY.
4230 152  DO 155 M = 1, DIR(3)
4240      IF(NSS.NE.SYSCOD(M)) GO TO 155
4250      SECNO = DESEC(M)
4260* - - IF THERE ARE DELETE MARKED RECORDS (AS SHOWN IN THE DIRECTORY),
4270*     CHOOSE STARTING SECTOR RATHER THAN ENDING.
4280      IF(NDEL(M).GT.0) SECNO = DSSEC(M)
4290      GO TO 170
4300 155  CONTINUE
4310* - - NO MATCH IN DIRECTORY.
4320*     CREATE NEW DIRECTORY ENTRY.
4330      IF(DIR(3).LT.40) GO TO 160
4340      PRINT 157, NSS
4350 157  FORMAT (1X, "NO MORE ROOM IN DIRECTORY FOR SYSTEM CODE ",
4360&             A2, ", RECORD IGNORED.")
4370 GO TO 200
4380* - - GET NEXT AVAILABLE SECTOR NUMBER.
4390 160  SECNO = DATAS(3)
4400      M = M + 1
4410      DATAS(3) = DATAS(3) + 1
4420      IF(DATAS(2).GT.(DATAS(3) - DATAS(1))) GO TO 165
4430 162  PRINT 164, DATAS(3)
4440 164  FORMAT (1X, "HAVE RUN OUT OF ROOM FOR DATA AT SECTOR ",
4450&             "NUMBER ", I4, /, "TERMINATING ADD ROUTINE.")
4460      GO TO 1000
4470 165  DSSEC(M) = SECNO
4480      DESEC(M) = SECNO
4490      DWSEC(M) = 1
4500      NDEL(M) = 0
4510      DIR(3) = DIR(3) + 1
4520      DIREUP = .T.
4530      PRINT 167, NSS
4540 167  FORMAT (1X, "PLEASE ENTER A SYSTEM NAME OF 12 OR FEWER ",
4550&             "CHARACTERS FOR SYSTEM CODE ", A2)
4560      READ, SYSNAM(M)
4570      SYSCOD(M) = NSS
4580 170  CALL RSEC (CARD, SECNO, CONT, $900)
```

```
4590* - - LOOK FOR NEXT AVAILABLE RECORD SLOT IN THE SECTOR.
4600 175  DO 180 NI = 1, 5
4610      DECODE (FREC(NI), 177) FG
4620 177  FORMAT (T2, A1)
4630      IF(FG.EQ.'C' .OR. FG.EQ.'S') GO TO 180
4640      IF(FG.EQ.'D') NDEL(M) = NDEL(M) - 1
4650      FREC(NI) = NEWR(IC)
4660      DATAUP = .T.
4670      OSS = NSS
4680      GO TO 200
4690 180  CONTINUE
4700* - - THIS SECTOR IS FULL.  CHECK TO SEE IF THERE IS ANOTHER
4710*     SECTOR BEYOND THIS ONE.
4720      IF(CONT.EQ.0) GO TO 190
4730* - - YES, THERE IS.  WRITE THIS SECTOR BACK AND GET NEXT.
4740      CALL WSEC (CARD, SECNO, CONT)
4750      SECNO = CONT; GO TO 170
4760* - - GET NEXT AVAILABLE SECTOR NUMBER.
4770 190  NSEC = DATAS(3)
4780      DATAS(3) = DATAS(3) + 1
4790      IF(DATAS(2).LT.(DATAS(3) - DATAS(1))) GO TO 162
4800      CONT = NSEC
4810      CALL WSEC (CARD, SECNO, CONT)
4820      CALL RSEC(CARD, NSEC, CONT, $900)
4830      DESEC(M) = NSEC
4840      SECNO = NSEC
4850      DNSEC(M) = DNSEC(M) + 1
4860      DIREUP = .T.
4870      GO TO 175
4880 200  CONTINUE
4890C
4900* - - HAVE FINISHED THOSE RECORDS.  ARE WE DONE?
4910 205  IF(.NOT. DONE) GO TO 82
4920 210  IF(TEST) GO TO 215
4930* - - WRITE DATA SECTOR BACK TO FILE.
4940      CALL WSEC (CARD, SECNO, CONT)
4950      CALL RWHEAD(2)
4960 215  PRINT, "ADD FUNCTION COMPLETE."
4970      IF(TEST) PRINT, "THIS HAS BEEN A TEST."
4980      GO TO 1000
4990C
5000 900  PRINT, "EOF ON SECTOR READ FOR ADREC."
5010      GO TO 990
5020 910  PRINT, "EOF ON ADREC READ OF TEMP FILE."
5030      GO TO 990
5040 920  PRINT, "ERROR RETURN FROM ADREC TEMP FILE READ."
5050      GO TO 990
5060 990  PRINT, "RETURNING TO MAIN PROGRAM."
5070 1000 RETURN
5080      END
```

```
1000CCRITER   BUILD SELECTION CRITERIA TABLES.
1010      SUBROUTINE CRITER
1020C        LATEST UPDATE 2 APRIL 79
1030      CHARACTER  INPUT*80, HOLD*6(5), BLANKS*6/'      '/, ANS*1,
1040&              TEMP*1, QUOTE*1/1H'/, CVAL*6(25),TEMP2*6,
1050&              MFGCOD*8(10), DRFC*2(18), COMP*2(6),TEMP3*6,
1060&              DTEMP*2, LHOLD*30, CTEXT*30(5), ZEROS*6/6H000000/
1070C
1080      INTEGER    FUNC(25), IVAL(25), MPL(25), FP, LEVEL(15),
1090&              OUTCOM(15), LOP(15), MFGPT, MP, OPARIN, CPARIN,
1100&              OCOMP, PTEXT, NCHA, ODRFC, LEADN, OMINF, KLOP,
1110&              SWITS(15), OROTAT
1120C
1130      LOGICAL    LIT, MAJF, INWK, MINF, ILIT, ALIT, COMPF, BY1,
1140&     BY2, BY3, MOSTR, DIREUP, INFOUP, DATAUP, WINFO, RTN
1150C
1160      COMMON /SWITS/ LIT, MAJF, INWK, MINF, ILIT, ALIT, COMPF,
1170&               BY1, BY2, BY3, MOSTR, RTN, DUM13, DUM14, DUM15
1180C
1190      EQUIVALENCE    (LHOLD, HOLD(1), IROTAT),
1200&                    (SWITS(1), LIT),(TEMP,TEMP2),(DTEMP,TEMP3),
1210&                    (HOLD(6), OROTAT)
1220C
1230      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
1240C
1250      COMMON /TABLES/ FUNC, CVAL, IVAL, MPL, LEVEL, OUTCOM,
1260&                     LOP, MFGCOD, CTEXT
1270C
1280      DATA       DRFC/'1 ', '2 ', '3 ', '4 ', '5 ', '6 ',
1290&                   '7 ', '8 ', '9 ', '10', '11', '12',
1300&                   '13', '14', '15', '16', '17', '18'/,
1310&                COMP/'EQ', 'NE', 'GE', 'LE', 'GT', 'LT'/
1320C
1330*****************************************************************
1340*
1350      IF(.NOT. WINFO) GO TO 4
1360      PRINT, "INSTRUCTIONS (Y OR N)?"
1370      READ, ANS
1380      IF(ANS.EQ.'Y') GO TO 2
1390      WINFO = .F.; GO TO 4
1400 2    PRINT, "A RECORD WILL BE SELECTED IF THE VALUES FOR SPECIFIED FIEL",
1410&           "DS MEET YOUR SELECT"
1420      PRINT, "CRITERIA IN SUCH A WAY AS TO MAKE THE OVERALL CRITERIA STR",
1430&           "ING LOGICALLY TRUE."
1440      PRINT, "FIELD NUMBERS ARE DESIGNATED AS F1, F2, F3...F18 FOR FIELD",
1450&           "S 1, 2, 3, AND 18"
1460      PRINT, "RESPECTIVELY.  COMPARISON VALUES ARE EQ, NE, LT, GT, GE, L",
```

```
1470&           "E.  THE MINOR"
1480      PRINT, "FUNCTION LINK 'OR' MAY BE USED IF IT IS NOT NECESSARY TO R",
1490&           "EPEAT THE FIELD"
1500      PRINT, "NUMBER.  FOR EXAMPLE, F1 EQ 'A' OR 'B' OR 'D'$   MEANS:"
1510      PRINT, "SELECT ALL RECORDS WITH FIELD 1 EQUAL TO 'A' OR 'B' OR 'D'"
1520      PRINT, "(NOTE: '$' DENOTES THE END OF THE CRITERIA STRING.)"
1530      PRINT, "MAJOR FUNCTION LINKS OF 'AND' AND 'OR' ARE USED BETWEEN LO",
1540&           "GICALLY INDEPENDENT"
1550      PRINT, "STRINGS AND ARE DELIMITED BY COMMAS.  FOR EXAMPLE, F1 EQ '",
1560&           "A', AND, F2 EQ 'HD'$"
1570      PRINT, "MEANS: SELECT ALL RECORDS WITH FIELD 1 EQUAL TO 'A' AND FI",
1580&           "ELD 2 EQUAL TO 'HD'."
1590      PRINT, "DO YOU WANT A LISTING OF THE FIELD NUMBERS, THEIR FORMATS ",
1600&           "AND CONTENTS (Y OR N)"
1610      READ, ANS
1620      IF(ANS.EQ.'N') GO TO 3
1630      PRINT, "THE CONTENTS OF RECORD FIELDS 1 THRU 18 ARE AS FOLLOWS:"
1640      PRINT, "FIELD   FIELD   FIELD"
1650      PRINT, "NUMBER  FORMAT  CONTENTS"
1660      PRINT, "  F1      A1    SIMULATOR CODE"
1670      PRINT, "  F2      A2    FUNCTION CODE"
1680      PRINT, "  F3      A2    SUBSYSTEM CODE"
1690      PRINT, "  F4      A8    MFG MODULE CODE"
1700      PRINT, "  F5      A1    PROGRAM LANGUAGE"
1710      PRINT, "  F6      F6.2  TIMING - MFG EST"
1720      PRINT, "  F7      I5    INSTR - MFG EST"
1730      PRINT, "  F8      I5    DATA - MFG EST"
1740      PRINT, "  F9      F5.1  CPS - CURRENT"
1750      PRINT, "  F10     F6.2  TIMING - CURRENT"
1760      PRINT, "  F11     I5    INSTR - CURRENT"
1770      PRINT, "  F12     I5    DATA - CURRENT"
1780      PRINT, "  F13     I2    YEAR OF UPDATE"
1790      PRINT, "  F14     A2    COMPUTER SYSTEM"
1800      PRINT, "  F15     A30   COMMENTS TEXT"
1810      PRINT, "  F16     A1    USER FLAG 1"
1820      PRINT, "  F17     A1    USER FLAG 2"
1830      PRINT, "  F18     A1    USER FLAG 3"
1840      PRINT, " "
1850      PRINT, " "
1860 3    PRINT, "VALUES SPECIFIED FOR ALPHA FIELDS F1 THRU F5, AND F14 THRU",
1870&           " F18 MUST BE DELIM-"
1880      PRINT, "ITED BY QUOTES.  VALUES SPECIFIED FOR NUMERIC FIELDS F6 TH",
1890&           "RU F13 MUST BE"
1900      PRINT, "DELIMITED BY '/'.  FOR EXAMPLE,"
1910      PRINT, "  (F11 GE /250/, AND, F10 LE /5.05/), OR, F13 EQ /76/$"
1920      PRINT, "MORE EXAMPLES:"
1930      PRINT, "    F1 NE 'A'$"
1940      PRINT, "    F1 GT 'E'$"
1950      PRINT, "    F12 LT F8$"
1960      PRINT, "UP TO 80 CHARACTERS MAY BE ENTERED PER LINE.  LONG LINES ",
1970&           "MAY BE CONTINUED ON"
1980      PRINT, "FOLLOWING LINES.  SIMSIZ WILL PROMPT WITH A '=' UNTIL '$' ",
```

```
1990&          "IS DETECTED."
2000      PRINT, "(SEE USER'S GUIDE FOR ADDITIONAL INFORMATION.)"
2010 4    PRINT, "INPUT STRING NOW.  ENTER 'I' FOR INSTRUCTIONS."
2020*
2030* - - CLEAR LOGIC TABLES.
2040 5    DO 7 I = 1, 25
2050      FUNC(I) = 0
2060      IVAL(I) = 0
2070      CVAL(I) = BLANKS
2080      MPL(I) = 0
2090 7    CONTINUE
2100      DO 8 I = 1, 15
2110      LEVEL(I) = 0
2120      OUTCOM(I) = 0
2130      LOP(I) = 0
2140      SWITS(I) = 0
2150 8    CONTINUE
2160      MP = 0
2170      FP = 1
2180      OMINF = 0
2190      PTEXT = 1
2200      MFCPT = 1
2210      OPARIN = 0
2220      CPARIN = 0
2230      ODRFC = 0
2240      IDRFC = 0
2250      OCOMP = 0
2260      TEMP2 = BLANKS
2270      TEMP3 = BLANKS
2280 10   READ 610, INPUT
2290      IF(INPUT.EQ.'I') GO TO 2
2300      PRINT 612, INPUT
2310 610  FORMAT (A80)
2320 612  FORMAT (1X, A80)
2330* - - BEGIN DECODING INPUT.
2340      IP = 1
2350      IF(MP.GT.0) GO TO 15
2360 12   NCHA = 0
2370      LEADN = 0
2380      LHOLD = " "
2390* - - EXTRACT NEXT CHARACTER.
2400 15   IF(IP.GT.80) GO TO 10
2410      CALL CONCAT (TEMP, 1, INPUT, IP, 1)
2420* - - FIND OUT WHICH CHARACTER IT IS.
2430      IF(TEMP.EQ.' ') GO TO 40
2440      IF(TEMP.EQ.'$') GO TO 500
2450      IF(TEMP.EQ.'(') GO TO 400
2460      IF(TEMP.EQ.')') GO TO 430
2470      IF(TEMP.EQ.',') GO TO 475
2480      NOSTR = .F.
2490      IF(TEMP.EQ.QUOTE .OR. TEMP.EQ.'"') GO TO 300
2500      IF(TEMP.EQ.'/') GO TO 200
```

```
2510      IF(TEMP.EQ.'.') GO TO 230
2520*
2530* - - NOT A CONTROL CHARACTER.
2540      IF(LIT) GO TO 70
2550      IF(MAJF) GO TO 70
2560      IF(MINF) GO TO 260
2570      IF(COMPF) GO TO 50
2580C
2590* - - MUST BE A DRFC CODE.
2600      IF(TEMP.EQ.'F') GO TO 26
2610 24   PRINT, "FIELD NUMBERS MUST BE PRECEEDED BY 'F'."
2620      GO TO 33
2630 26   IP = IP + 1
2640      CALL CONCAT (DTEMP, 1, INPUT, IP, 2)
2650 28   DO 30 IDRFC = 1, 18
2660      IF(DTEMP.EQ.DRFC(IDRFC)) GO TO 35
2670 30   CONTINUE
2680      IF(RTN) GO TO 31
2690      RTN = .T.
2700* - - OVERWRITE 2ND CHARACTER IN DTEMP WITH A BLANK.
2710      CALL CONCAT (DTEMP, 2, BLANKS, 1, 1)
2720      IP = IP - 1
2730      GO TO 28
2740* - - NOT AN ACCEPTABLE DRFC VALUE.
2750 31   PRINT 32, DTEMP
2760 32   FORMAT (1X, A2, " IS NOT A VALID DATA RECORD FIELD CODE.")
2770      RTN = .F.
2780 33   PRINT 34
2790 34   FORMAT (1X, "RE-INPUT CONTROL CHARACTER STRING.")
2800* - - RESET LOGIC TABLE.
2810      GO TO 5
2820C
2830* - - HAVE A VALID DRFC.
2840 35   FUNC(FP) = 4
2850      IVAL(FP) = IDRFC
2860* - - SAVE THIS IDRFC.
2870      ODRFC = IDRFC
2880      MP = MP + 1
2890      MPL(FP) = MP
2900      LEVEL(MP) = OPARIN
2910      COMPF = .T.
2920      IP = IP + 2
2930      FP = FP + 1
2940      GO TO 15
2950C
2960* - - CHARACTER IS A BLANK.
2970 40   IF(ALIT) GO TO 70
2980      IF(INWK) GO TO 80
2990      IP = IP + 1
3000      GO TO 15
3010C
3020* - - EXTRACT COMPARISON FUNCTION CODE.
```

```
3030 50   CALL CONCAT (DTEMP, 1, INPUT, IP, 2)
3040      DO 55 ICOMP = 1, 6
3050      IF(DTEMP.EQ.COMP(ICOMP)) GO TO 65
3060 55   CONTINUE
3070      PRINT 60, DTEMP
3080 60   FORMAT (1X, A2, " IS NOT A VALID COMPARISON CODE.")
3090      GO TO 33
3100C
3110* - - HAVE A VALID COMP VALUE.
3120 65   IF(ICOMP.EQ.OCOMP) GO TO 67
3130      FUNC(FP) = 3
3140      IVAL(FP) = ICOMP
3150* - - SAVE THIS ICOMP VALUE.
3160      OCOMP = ICOMP
3170      MPL(FP) = MP
3180      FP = FP + 1
3190 67   COMPF = .F.
3200      LIT = .T.
3210      IP = IP + 2
3220      GO TO 12
3230C
3240* - - WORK LITERAL OR MAJF VALUE.
3250 70   INWK = .T.
3260      IP = IP + 1
3270      IF(NCHA.EQ.30) GO TO 72
3280      NCHA = NCHA + 1
3290      CALL CONCAT (LHOLD, NCHA, TEMP, 1, 1)
3300      GO TO 15
3310* - - ERROR - CHARACTER STRING TOO LONG.
3320 72   PRINT 75, LHOLD
3330 75   FORMAT (1X, "THE FOLLOWING LITERAL CHARACTER STRING HAS ",
3340&               "BEEN TRUNCATED AT 30 CHARACTERS: ", /, A30)
3350      GO TO 15
3360C
3370* - - HAVE ENCOUNTERED A BLANK IN LITERAL STRING.
3380 80   IF(ILIT) GO TO 84
3390      IF(MAJF) GO TO 87
3400      IF(NCHA.GT.1) GO TO 81
3410      IF(ODRFC.EQ.1 .OR. ODRFC.EQ.5 .OR. ODRFC.GT.15) GO TO 100
3420 81   CALL CONCAT (TEMP, 1, LHOLD, 1, 1)
3430      IF(TEMP.EQ.'F') GO TO 120
3440      PRINT 82,  LHOLD
3450 82   FORMAT (1X, "CANNOT PROCESS FOLLOWING PORTION OF INPUT ",
3460&            "STRING:", /, A30)
3470      GO TO 33
3480 84   PRINT 85
3490 85   FORMAT (1X, "BLANK ENCOUNTERED IN NUMERIC LITERAL STRING.",
3500&            / , "IGNORE BLANK, CONTINUE PROCESSING.")
3510 87   IP = IP + 1
3520      GO TO 15
3530C
3540* - - STORE NUMERIC LITERAL.
```

```
3550 90   IF(NCHA.LT.6) GO TO 95
3560 91   PRINT 92, LHOLD
3570 92   FORMAT (1X, "WARNING: TOO MANY CHARACTERS IN FOLLOWING ",
3580&               "NUMERIC LITERAL: ", /, A30)
3590      GO TO 33
3600 95   IF(ODRFC.EQ.6 .OR. ODRFC.EQ.9 .OR. ODRFC.EQ.10) GO TO 100
3610* - - IF DRFC IS FOR YR, DON'T SHIFT NUMBERS.
3620      IF(ODRFC.EQ.13) GO TO 100
3630* - - JUST STORE A 5 DIDGET VALUE.
3640      IF(NCHA.EQ.5) GO TO 100
3650* - - FILL THE REST OF THE HOLD WORD WITH ZEROS.
3660      CALL CONCAT (HOLD(1), NCHA+1, ZEROS, 1, 6-NCHA)
3670      OROTAT = ILR (IROTAT, (6*(NCHA+1)))
3680      HOLD(1) = HOLD(6)
3690      NCHA = 5
3700      GO TO 100
3710C
3720 100  IF(NCHA.EQ.5) GO TO 105
3730* - - FILL WITH TRAILING ZEROES.
3740      CALL CONCAT (HOLD(1), (NCHA+1), ZEROS, 1, (5-NCHA))
3750* - - CHECK FOR PROPER DECIMAL POINT POSITION.
3760 105  J = 3 - LEADN
3770      IF(J.EQ.3) GO TO 115
3780      IF(J.EQ.0) GO TO 100
3790* - - INSERT LEADING ZEROS.
3800      CALL CONCAT (HOLD(6), 1, ZEROS, 1, J)
3810* - - MOVE THE REST OF THE CHARACTERS.
3820      CALL CONCAT (HOLD(6), (J+1), HOLD(1), 1, (6-J))
3830* - - REPLACE VALUE.
3840      HOLD(1) = HOLD(6)
3850      NCHA = 5
3860* - - IF MORE THAN 5 CHARACTERS, TRUNCATE TO 5.
3870 100  IF(NCHA.GT.5) NCHA = 5
3880* - - MOVE CHARACTERS INTO TARGET VALUE.
3890 110  CALL CONCAT (CVAL(FP), 1, LHOLD, 1, NCHA)
3900      FUNC(FP) = 5
3910      IF(OMINF.EQ.2) FUNC(FP) = -5
3920      IF(OMINF.EQ.3) FUNC(FP) = -7
3930 112  MPL(FP) = MP
3940      FP = FP + 1
3950      IF(BY3) GO TO 505
3960      LIT = .F.
3970      INNK = .F.
3980      IF(BY1 .OR. BY2) GO TO 445
3990      MINF = .T.
4000      IP = IP + 1
4010      GO TO 12
4020C
4030 115  PRINT 117, HOLD(1)
4040 117  FORMAT (1X, "NO DECIMAL PT IN FOLLOWING NUMERIC LITERAL: ",/,A6)
4050      GO TO 33
4060* - - HAVE TWO-CHARACTER LITERAL.  CHECK TO SEE IF IT
```

```
4070*     IS A DRFC VALUE.
4080 120  CALL CONCAT (DTEMP, 1, HOLD(1), 2, 2)
4090      DO 130 IDRFC = 1, 15
4100      IF(DTEMP.EQ.DRFC(IDRFC)) GO TO 135
4110 130  CONTINUE
4120* - - NOT A DRFC VALUE.
4130      GO TO 31
4140* - - HAVE A VALID DRFC AS A TARGET LITERAL.
4150 135  IVAL(FP) = IDRFC
4160      FUNC(FP) = -6
4170      IF(OMINF.EQ.0) FUNC(FP) = 6
4180      GO TO 112
4190C
4200* - - SET ILIT.
4210 200  IF(ILIT) GO TO 220
4220      IF(LIT) GO TO 210
4230* - - ERROR - NOT EXPECTING LITERAL.
4240 202  PRINT 205, IP
4250 205  FORMAT (1X, "UNEXPECTED LITERAL DELIMITER ENCOUNTERED ",
4260&            "WHILE PROCESSING CONTROL STRING.", /,
4270&            "CHARACTER IS ", I2, "TH CHARACTER IN CONTROL STRING.")
4280      GO TO 33
4290* - - BEGINNING OF NUMERIC LITERAL.
4300 210  ILIT = .T.
4310      GO TO 87
4320C
4330* - - END OF NUMERIC LITERAL.
4340 220  ILIT = .F.
4350      LIT = .F.
4360      GO TO 90
4370C
4380* - - HAVE FOUND DECIMAL POINT.
4390 230  IF(ALIT) GO TO 70
4400      IF(ILIT) GO TO 240
4410      PRINT 235, LHOLD
4420 235  FORMAT (1X, "UNEXPECTED PERIOD AFTER FOLLOWING STRING: ",/,
4430&            A30)
4440      GO TO 33
4450 240  LEADN = NCHA
4460      IF(LEADN.LE.3) GO TO 87
4470      PRINT 245, HOLD(1)
4480 245  FORMAT (1X, "ERROR IN DECIMAL POINT PLACEMENT FOLLOWING ",
4490&                "NUMERIC LITERAL: ", A6)
4500      GO TO 33
4510C
4520* - - EXTRACT MINOR FUNCTION.
4530 260  CALL CONCAT (DTEMP, 1, INPUT, IP, 2)
4540      IF(DTEMP.EQ.'OR') GO TO 270
4550      IF(DTEMP.EQ.'TO') GO TO 280
4560      PRINT 265, DTEMP
4570 265  FORMAT (1X, A2, " IS AN IMPROPER VALUE FOR A MINOR FUNCTION.")
4580      GO TO 33
```

```
4590* - - HAVE FOUND MINF = 2.
4600 270  IF(OMINF.EQ.2) GO TO 275
4610* - - SET UP ENTRY IN FUNC TABLE.
4620      OMINF = 2
4630      FUNC(FP) = -2
4640      IVAL(FP) = 2
4650      MPL(FP) = MP
4660 274  FP = FP + 1
4670 275  MINF = .F.
4680      LIT = .T.
4690      IP = IP + 2
4700      GO TO 12
4710* - - HAVE FOUND MINF = 3.
4720 280  IF(OMINF.EQ.3) GO TO 275
4730      OMINF = 3
4740      FUNC(FP) = 2
4750      IVAL(FP) = 3
4760      MPL(FP) = MP
4770      GO TO 274
4780C
4790* - - FOUND A QUOTE.
4800 300  IF(ALIT) GO TO 307
4810      IF(LIT) GO TO 305
4820* - - ERROR NOT EXPECTING A QUOTE.
4830      GO TO 202
4840* - - SET UP FOR ALPHA LITERAL STRING.
4850 305  ALIT = .T.
4860      GO TO 87
4870* - - END OF ALIT STRING.
4880 307  ALIT = .F.
4890      LIT = .F.
4900* - - CHECK FOR MANUFACTURER'S CODE.
4910      IF(ODRFC.NE.4) GO TO 315
4920* - - HAVE MFG'S MODULE CODE.
4930      IF(MFGPT.LE.10) GO TO 310
4940      PRINT 308, LHOLD
4950 308  FORMAT (1X, "NO MORE ROOM IN TABLES FOR FOLLOWING MODULE ",
4960&           "CODE: ",/, A30, /, "LIMIT OF 10 CODES HAS BEEN EXCEEDED.",
4970&           "  FURTHER MODULE CODES WILL BE IGNORED.")
4980* - - IF A CONTINUING STRING OF ORS - SKIP THIS ENTRY.
4990      IF(FUNC(FP - 1).EQ.-5) GO TO 314
5000* - - REINSERT LAST VALUE IN TABLE.
5010      LHOLD = MFGCOD(10)
5020      MFGPT = 10
5030 310  CALL CONCAT (MFGCOD(MFGPT), 1, LHOLD, 1, 8)
5040      IVAL(FP) = MFGPT
5050      MFGPT = MFGPT + 1
5060 312  FUNC(FP) = -5
5070      IF(OMINF.EQ.0) FUNC(FP) = 5
5080      MPL(FP) = MP
5090      FP = FP + 1
5100 314  IP = IP + 1
```

```
5110      INNK = .F.
5120      MINF = .T.
5130      GO TO 12
5140* - - CHECK FOR CTEXT STRING.
5150 315  IF(ODRFC.EQ.15) GO TO 325
5160* - - HAVE LIT BUT NOT FOR FIELD 4 OR 15.
5170      GO TO 108
5180* - - HAVE A CTEXT STRING.  CHECK FOR ROOM IN CTEXT TABLE.
5190 325  IF(PTEXT.LE.5) GO TO 335
5200      PRINT 330, LHOLD
5210 330  FORMAT (1X, "NO MORE ROOM IN TABLES FOR FOLLOWING TEXT ",
5220&             "STRING: ", /, A30, /, "LIMIT OF 5 ENTRIES HAS BEEN ",
5230&             "EXCEEDED.  THIS ENTRY IS BEING DISPOSED OF.")
5240      GO TO 314
5250 335  CALL CONCAT (CTEXT(PTEXT), 1, LHOLD, 1, 30)
5260      IVAL(FP) = PTEXT
5270      PTEXT = PTEXT + 1
5280      GO TO 312
5290C
5300* - - HAVE FOUND '('.  INCREMENT LEVEL OF PARENTHESIS.
5310 400  OPARIN = OPARIN + 1
5320      IF(COMPF .OR. LIT .OR. MINF .OR. MAJF) GO TO 404
5330      GO TO 87
5340 404  PRINT 406, IP
5350 406  FORMAT (1X, "UNEXPECTED PARENTHESIS ENCOUNTERED IN ",
5360&           I2, "TH CHARACTER POSITION OF THE ", /,
5370&           "INPUT CONTROL STRING.")
5380      GO TO 33
5390C
5400* - - HAVE FOUND ')'.
5410 430  CPARIN = CPARIN + 1
5420      IF(CPARIN.LE.OPARIN) GO TO 440
5430      PRINT 435, IP
5440 435  FORMAT (1X, "PARENTHESIS AT CHARACTER POSITION ", I2,
5450&           " DOES NOT BALANCE.")
5460      GO TO 33
5470 440  IF(MOSTR) GO TO 449
5480      IF(COMPF .OR. ALIT .OR. MAJF .OR. ILIT) GO TO 404
5490      IF(.NOT. LIT) GO TO 445
5500      BY1 = .T.
5510      GO TO 80
5520* - - SEE IF NEED TO INDICATE END OF FUNCTION.
5530 445  IF(OMINF.EQ.3) GO TO 447
5540      OMINF = 0
5550      FUNC(FP) = -2
5560      IVAL(FP) = 0
5570      MPL(FP) = MP
5580      IF(FUNC(FP-1).EQ.-5 .OR.
5590&        FUNC(FP-1).EQ.-6) FUNC(FP-1) = -FUNC(FP-1)
5600      FP = FP + 1
5610      GO TO 449
5620 447  IF(FUNC(FP-1).EQ.5 .OR.
```

```
5630&       FUNC(FP-1).EQ.6) FUNC(FP-1) = -FUNC(FP-1)
5640 449  IF(BY2) GO TO 485
5650      MP = MP + 1
5660      LEVEL(MP) = OPARIN - 1
5670      FUNC(FP) = -1
5680      MPL(FP) = MP
5690      FP = FP + 1
5700      MINF = .F.
5710      IP = IP + 1
5720      BY1 = .F.
5730      MOSTR = .T.
5740      OPARIN = OPARIN - 1
5750      CPARIN = CPARIN - 1
5760      GO TO 12
5770C
5780* - - HAVE FOUND A COMMA.
5790 475  IP = IP + 1
5800      IF(MAJF) GO TO 490
5810      IF(MOSTR) GO TO 485
5820      IF(COMPF .OR. ILIT) GO TO 478
5830      GO TO 482
5840 478  PRINT 480, IP
5850 480  FORMAT (1X, "UNEIPECTED COMMA AT CHARACTER POSITION ", I2)
5860      GO TO 33
5870 482  IF(ALIT) GO TO 70
5880      BY2 = .T.
5890      IF(.NOT. LIT) GO TO 445
5900* - - POSSIBLE LITERAL FOLLOWED BY COMMA.
5910      GO TO 80
5920 485  MINF = .F.
5930      MAJF = .T.
5940      BY2 = .F.
5950      GO TO 12
5960C
5970* - - HAVE FOUND END OF MAJF.
5980 490  IF(HOLD(1).EQ.'AND') GO TO 494
5990      IF(HOLD(1).EQ.'OR') GO TO 496
6000      PRINT 492, HOLD(1), IP - 1
6010 492  FORMAT (1X, "MAJOR FUNCTION CODE ", A6, " ENDING IN ",
6020&           "CHARACTER POSITION ", I2, /, "IS NOT A PROPER ",
6030&           "MAJOR FUNCTION CODE.")
6040      GO TO 33
6050 494  KLOP = 1
6060      GO TO 497
6070 496  KLOP = 2
6080 497  LOP(MP) = KLOP
6090      INWK = .F.
6100      MAJF = .F.
6110      GO TO 12
6120C
6130* - - FINI...
6140 500  LOP(MP) = 0
```

```
6150      IF(NOSTR .OR. NCHA.EQ.0) GO TO 505
6160      BY3 = .T.
6170      GO TO 80
6180 505  FUNC(FP) = -8
6190* - - INSURE THAT PRECEEDING FUNC IS NEGATIVE.
6200      IF(FUNC(FP-1).GT.0) FUNC(FP-1) = -FUNC(FP-1)
6210      IF(OPARIN.EQ.CPARIN) GO TO 515
6220      PRINT 510
6230 510  FORMAT (1X, "PARENTHESIS DO NOT BALANCE.")
6240      GO TO 33
6250 515  RETURN
6260      END
```

SUBROUTINE DARAY

```
1000CDARAY   DISPLAY 'ARRAYS' COMMON.
1010C
1020       SUBROUTINE DARAY (MODE)
1030C
1040C        MODE:
1050C          1 = PRINT ALL VALUES.
1060C          2 = PRINT RECORDS ONLY.
1070C
1080       CHARACTER  STR*8(15), LSTR*30, OUT*70, FORM1*12/"(T  ,I4,',')"/,
1090&                 FORM2*12/"(T  ,' TO ')"/, FORM3*8/'(1X,A  )'/,
1100&                 NULL*30/'*NULL*'/
1110C
1120       INTEGER  RP, FP, SP, REC(20), FLD(15), OP
1130C
1140       COMMON /ARRAYS/ RP, REC, FP, FLD, SP, STR, LSTR, NFLDS
1150C
1160       IF(MODE.EQ.1) GO TO 3
1170       PRINT, "RECORDS TO BE DELETED:"
1180       GO TO 4
1190 3     PRINT, "RECORDS TO BE MODIFIED:"
1200 4     OP = 1
1210       DO 20 I = 1, RP
1220       ENCODE (FORM1, 5) OP
1230 5     FORMAT (T3, I2)
1240       NUM = REC(I)
1250       IF(NUM.GE.0) GO TO 10
1260       NUM = -NUM; OP = OP - 1
1270       ENCODE (FORM2, 5) OP
1280       ENCODE (OUT, FORM2)
1290       OP = OP + 4
1300       ENCODE (FORM1, 5) OP
1310 10    ENCODE (OUT, FORM1) NUM
1320       OP = OP + 5
1330       IF(OP.GT.60) GO TO 22
1340 20    CONTINUE
1350       OP = OP - 1; IF(OP.LT.2) GO TO 27
1360 22    ENCODE (FORM3, 24) OP - 1
1370 24    FORMAT (T6,I2)
1380       WRITE (06, FORM3) OUT
1390       OP = 1
1400       IF(I.LT.RP) GO TO 20
1410 27    IF(MODE.EQ.2) RETURN
1420       PRINT, "FIELDS TO BE MODIFIED AND NEW VALUES:"
1430       DO 100 I = 1, FP
1440       IF(FLD(I).EQ.15) GO TO 50
1450 28    WRITE (06, 30) FLD(I), STR(I)
1460 30    FORMAT (1X, I3, " = ", A8)
```

```
1470      GO TO 100
1480 50   IF(LSTR.EQ.NULL) GO TO 28
1490      WRITE (06, 60) FLD(I), LSTR
1500 60   FORMAT (1X, I3, " = ", A30)
1510 100  CONTINUE
1520      RETURN; END
```

SUBROUTINE DIRMOD

```
      CDIRMOD   DIRECTORY MODIFIER ROUTINE.
C
      SUBROUTINE DIRMOD
C
      CHARACTER  FNAME*6, DATES*8(3), SYSCOD*2(40), SYSNAM*12(40),
     &           ANS*1, SNAME*12, SCODE*2
C
      INTEGER  SECNO, LSIZE, DIR(3), INFO(3), DATAS(3), DSSEC(40),
     &         DESEC(40), DNSEC(40), NDEL(40), RNUM
C
      LOGICAL  DIREUP, INFOUP, DATAUP, WINFO
C
      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
C
      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
C
      COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DNSEC, NDEL
C
C
      IF(.NOT. WINFO) GO TO 40
      PRINT, "INSTRUCTIONS (Y OR N)?"
      READ, ANS
      IF(ANS.EQ.'Y') GO TO 25
      WINFO = .F.; GO TO 40
   25 WINFO = .T.
      PRINT, "THIS ROUTINE WILL ALLOW ONE TO MODIFY THE DATA BASE"
      PRINT, "DIRECTORY AS FOLLOWS:"
      PRINT, "  A - ADD NEW SYSTEM NAMES AND CODES TO THE DIRECTORY."
      PRINT, "  C - CHANGE SYSTEM NAMES IN THE DIRECTORY."
      PRINT, "  D - DELETE OBSOLETE OR INCORRECT DIRECTORY ENTRIES."
      PRINT, "  L - LIST DIRECTORY CONTENTS."
      PRINT, " "
   40 PRINT, "ENTER: A (ADD), C (CHANGE), D (DELETE), E (END), I (",
     &           "INSTRUCTIONS), L (LIST)."
      READ, ANS
      IF(ANS.EQ.'A') GO TO 60
      IF(ANS.EQ.'C') GO TO 150
      IF(ANS.EQ.'D') GO TO 200
      IF(ANS.EQ.'E') GO TO 260
      IF(ANS.EQ.'I') GO TO 25
      IF(ANS.EQ.'L') CALL DRLIST
      GO TO 40
C
C - - ADD NEW DIRECTORY ENTRY.
   60 PRINT, "ENTER NEW SYSTEM NAME OF 12 OR FEWER CHARACTERS (IN ",
     &       "QUOTES),"
      PRINT, "AND THE TWO-CHARACTER SYSTEM CODE."
```

```
1470      READ, SNAME, SCODE
1480* - - CHECK FOR DUPLICATE SYSCOD VALUE.
1490      DO 70 I1 = 1, DIR(3)
1500      IF(SCODE.EQ.SYSCOD(I1)) GO TO 80
1510 70   CONTINUE
1520      GO TO 90
1530* - - THAT CODE IS ALREADY IN USE.
1540 80   PRINT 85, SCODE, SYSNAM(I1)
1550 85   FORMAT (1X, "'", A2, "' IS ALREADY IN USE FOR '", A12,"'.")
1560      GO TO 40
1570* - - CREATE NEW DIRECTORY ENTRY.
1580 90   IF(DIR(3).LT.40) GO TO 100
1590      PRINT 95, DIR(3)
1600 95   FORMAT (1X, "THERE ARE CURRENTLY ", I2, " ENTRIES IN THE "
1610&     "DIRECTORY.",/, "NO MORE ROOM FOR NEW ENTRIES.")
1620      GO TO 40
1630* - - GET NEXT AVALIABLE SECTOR NUMBER.
1640 100  SECNO = DATAS(3)
1650      DATAS(3) = DATAS(3) + 1
1660      IF(DATAS(2).GT.(DATAS(3)-DATAS(1))) GO TO 110
1670      PRINT, "CANNOT CREATE NEW DIRECTORY ENTRY DUE TO LACK OF SPACE"
1680      PRINT, "FOR DATA RECORDS.  RECOMMEND REBUILDING THE DATA"
1690      PRINT, "BASE ON LARGER FILE SPACE."
1700      DATAS(3) = DATAS(3) - 1
1710      GO TO 40
1720* - - INSERT VALUES IN DIRECTORY.
1730 110  DIR(3) = DIR(3) + 1
1740      M = DIR(3)
1750      DSSEC(M) = SECNO
1760      DESEC(M) = SECNO
1770      DNSEC(M) = 1
1780      NDEL(M) = 0
1790      DIREUP = .T.
1800      SYSCOD(M) = SCODE
1810      SYSNAM(M) = SNAME
1820* - - ADD COMPLETE.
1830      PRINT, "ADD COMPLETE."
1840      GO TO 40
1850C
1860* - - CHANGE A DIRECTORY SYSTEM NAME.
1870 150  PRINT, "ENTER NEW SYSTEM NAME (IN QUOTES) OF 12 OR FEWER"
1880      PRINT, "CHARACTERS, AND THE 'OLD' TWO-CHARACTER SYSTEM CODE."
1890      READ, SNAME, SCODE
1900      DO 160 I2 = 1, DIR(3)
1910      IF(SCODE.EQ.SYSCOD(I2)) GO TO 170
1920 160  CONTINUE
1930      PRINT 165, SCODE
1940 165  FORMAT (1X, "NO MATCH ON SYSTEM CODE '", A2, "'.")
1950      GO TO 40
1960* - - INSERT NEW SYSTEM NAME.
1970 170  SYSNAM(I2) = SNAME
1980      DIREUP = .T.
```

```
1990      PRINT, "NAME CHANGE COMPLETE."
2000      GO TO 40
2010C
2020* - - DELETE DIRECTORY ENTRY.
2030 200  PRINT, "ENTER THE TWO-CHARACTER SYSTEM CODE OF THE"
2040      PRINT, "DIRECTORY ENTRY TO BE DELETED."
2050      READ, SCODE
2060      DO 210 I = 1, DIR(3)
2070      IF(SCODE.EQ.SYSCOD(I)) GO TO 220
2080 210  CONTINUE
2090      GO TO 160
2100* - - RIPPLE ENTRIES UP IN DIRECTORY.
2110 220  L = I
2120      IF(L.EQ.DIR(3)) GO TO 250
2130      DO 240 J = L, DIR(3) - 1
2140      SYSNAM(J) = SYSNAM(J + 1)
2150      SYSCOD(J) = SYSCOD(J + 1)
2160      DSSEC(J) = DSSEC(J + 1)
2170      DESEC(J) = DESEC(J + 1)
2180      DNSEC(J) = DNSEC(J + 1)
2190      NDEL(J) = NDEL(J + 1)
2200 240  CONTINUE
2210* - - REDUCE THE NUMBER OF DIRECTORY ENTRIES IN IDSEC.
2220 250  DIR(3) = DIR(3) - 1
2230      DIREUP = .T.
2240      PRINT, "DELETE COMPLETE."
2250      GO TO 40
2260C
2270* - - WRITE ID AND DR COMMONS BACK TO THE DATA BASE.
2280 260  CALL RWHEAD (2)
2290      RETURN
2300      END
```

SUBROUTINE DRLIST

```
      C DRLIST   SUB TO DO FORMATED LIST OF DATA BASE DIRECTORY.
            SUBROUTINE DRLIST
      C
      C - - ROUTINE TO PERFORM FORMATED DUMP OF DATA BASE
      C     DIRECTORY SECTION.
      C
            CHARACTER  FNAME*6, DATES*8(3), SYSNAM*12(40), SYSCOD*2(40)
      C
            INTEGER  LSIZE, DIR(3), INFO(3), DATAS(3), RNUM,
           &         DSSEC(40), DESEC(40), DNSEC(40), NDEL(40)
      C
            COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
      C
            COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DNSEC, NDEL
      C
            WRITE (06, 100)
        100 FORMAT (1H0, 10X, "SIMSIZ DATA BASE DIRECTORY SECTION",//,
           &        22X, "DATA     DATA   SECTORS  NUMBER", /,
           &         4X, "SYSTEM    SYST  STARTING  ENDING   WITH",
           &        "    DELETED", /, 5X, "NAME     CODE   SECTOR",
           &        "   SECTOR   DATA    RECORDS",//)
            DO 150 I = 1, DIR(3)
            WRITE (06, 140) SYSNAM(I), SYSCOD(I), DSSEC(I), DESEC(I),
           &                DNSEC(I), NDEL(I)
        140 FORMAT (1X, A12, 2X, A2, 2(5X, I4), 5X, I3, 6X, I3)
        150 CONTINUE
            PRINT, " "
            RETURN
            END
```

```
1000CDSPLAY    DISPLAY RECORDS.
1010      SUBROUTINE DSPLAY (BUFF, NUMB, ITOTAL, STRT, POINT)
1020C
1030* - - WHERE:
1040*        BUFF = ADDRESS OF BUFFER CONTAINING RECORDS
1050*                 TO BE DISPLAYED.
1060*        NUMB = NUMBER OF RECORDS TO DISPLAY.
1070*        ITOTAL = NUMBER OF RECORDS DISPLAYED SO FAR.
1080*        STRT = VALUE TO START RENUMBERING RECORDS AT.
1090*        POINT = POINTER TO FIRST RECORD IN BUFF TO BE
1100*                DISPLAYED.
1110C
1120      CHARACTER  BUFF*450, REC*90(5), AC*1, SYS*2, SUB*2,
1130&                MODU*8, LAN*1, YR*2, SC*2, TEXT*30, FL1*1,
1140&                FL2*1, FL3*1, FL4*1, BUFF1*450
1150C
1160      INTEGER  OTOTAL, STRT, RECN, POINT
1170C
1180      EQUIVALENCE (BUFF1, REC(1))
1190C
1200      COMMON /TERMTYPE/ LINE
1210C
1220*****************************************************************
1230C
1240* - - MOVE DATA.
1250      BUFF1 = BUFF
1260C
1270* - - CHECK FOR ZERO RECORDS IN BUFFER.
1280      IF(NUMB.EQ.0) GO TO 70
1290      IF(ITOTAL.NE.0) GO TO 25
1300      OTOTAL = 0
1310      I = POINT
1320      RECN = STRT - 1
1330      GO TO 30
1340 25   I = 1
1350 30   DO 60 J = 1, NUMB
1360* - - CHECK FOR A BLANK RECORD.
1370      DECODE (REC(I), 32) FL1
1380 32   FORMAT (T2, A1)
1390      IF(FL1.EQ.' ') GO TO 70
1400      IF(LINE.EQ.80) IR = MOD(OTOTAL, 7)
1410      IF(LINE.GT.80) IR = MOD(OTOTAL, 20)
1420      IF(IR.NE.0) GO TO 40
1430      IF(LINE.EQ.80) GO TO 36
1440      PRINT 34
1450 34   FORMAT (1H0, " LINE S/C FUNC SUBS  MODULE      !--M",
1460&     "ANUFACTURER'S--!--------CURRENT---------!", 37X,
```

```
     "STAT USER FLAGS",/,"   NO      CODE CODE   NAME   ",
     "LANG TIMING INSTR  DATA  CPS  TIMING INSTR  DATA ",
     "YR CC", 14X, "TEXT", 14X, "FLAG  U1 U2 U3",//)
     GO TO 40
  36 PRINT 38
  38 FORMAT (1H0, " S/C FUNC SUBS  MODULE  LANG TIMING ",
             "INSTR  DATA  CPS  TIMING INSTR  DATA YR CC")
  40 OTOTAL = OTOTAL + 1
     ITOTAL = ITOTAL + 1
     RECN = RECN + 1
     DECODE (REC(I), 44) AC, SYS, SUB, MODU, LAN, TI1, IN1,
     DA1,CPS, TI2, IN2, DA2, YR, SC, TEXT, FL2, FL3, FL4
  44 FORMAT (A1, T3, 2A2, A8, A1, F5.2, 2I5, F4.1, F5.2, 2I5,
              2A2, A30, 3A1)
     IF(LINE.GT.80) GO TO 50
     PRINT 46, AC, SYS, SUB, MODU, LAN, TI1, IN1, DA1, CPS,
     TI2, IN2, DA2, YR, SC, RECN, TEXT, FL1, FL2, FL3, FL4
  46 FORMAT (1H0, 2X, A1, 2(3X, A2), 2X, A8, 3X, A1, F8.2,
             2I6, F6.1, F7.2, 2I6, 2(1X, A2), /, 1X,
             "REC#=", I3, "  TEXT=", A30, "  FLGS=", A1, 1X, 3A1)
     GO TO 54
  50 PRINT 52, RECN, AC, SYS, SUB, MODU, LAN, TI1, IN1,
             DA1, CPS, TI2, IN2, DA2, YR, SC, TEXT, FL1,
             FL2, FL3, FL4
  52 FORMAT (1X, I4, 3X, A1, 2(3X, A2), 2X, A8, 2X, A1,
             3X, F6.2, 2I6, F6.1, F7.2, 2I6, 2(1X, A2), 1X,
             A30, 2X, A1, 3X, 3(2X, A1))
  54 I = I + 1
  60 CONTINUE
  70 RETURN
     END
```

```
1000CEVAL    SUB TO SELECT RECORDS THAT MEET CRITERIA.
1010       SUBROUTINE EVAL (REC, ISTAT)
1020C
1030* - - WHERE:
1040*        REC = 90 CHARACTER RECORD TO BE EVALUATED.
1050*        ISTAT = RETURNED STATUS OF EVALUATION.
1060*           1 = REJECT RECORD.
1070*           2 = SELECT RECORD.
1080C
1090       CHARACTER  CVAL*6(25), MFGCOD*8(10), TARGET*8, TAR2*6,
1100&                 TEXT*30, CTEXT*30(5), REC*90
1110C
1120       INTEGER    FUNC(25), FP, IVAL(25), MINF, AFUN, OUTCOM(15),
1130&                 LEVEL(15), LOP(15), B, FRCB(20), NCBR(20),
1140&                 MPL(25), SUMER
1150C
1160       LOGICAL    EOT, SKIP, INCR, MAJF, SUMIT
1170C
1180       COMMON /TABLES/ FUNC, CVAL, IVAL, MPL, LEVEL, OUTCOM,
1190&                      LOP, MFGCOD, CTEXT
1200C
1210       DATA   FRCB/ 1,  3,  5,  7, 15, 16, 21, 26, 31, 35, 40,
1220&                45, 50, 52, 54, 84, 85, 86, 2*0/,
1230&             NCBR/ 1,  2,  2,  8,  1,  5,  5,  5,  4,  5,  5,
1240&                  5,  2,  2, 30,  1,  1,  1, 2*0/
1250*
1260*****************************************************************
1270*
1280 1     TARGET = ' '
1290       TAR2 = ' '
1300       TEXT = ' '
1310       FP = 0
1320       MAJF = .F.
1330       EOT = .F.
1340       SKIP = .F.
1350       INCR = .F.
1360       SUMIT = .F.
1370* - - RESET OUTCOM TABLE.
1380       DO 3 I = 1, 15
1390       OUTCOM(I) = 0
1400 3     CONTINUE
1410 5     FP = FP + 1
1420* - - TAKE THE ABSOLUTE VALUE OF THE NEXT FUNCTION.
1430       AFUN = ABS(FUNC(FP))
1440* - - UPDATE VALUES VIA THE FUNC TABLE.
1450       GO TO (10, 15, 20, 25, 30, 35, 40, 45), AFUN
1460* - - SET MAJF TRUE.
```

```
1470 10   MAJF = .T.
1480      GO TO 70
1490* - - SET MINOR FUNC CODE.
1500 15   MINF = IVAL(FP)
1510      GO TO 70
1520* - - SET ICOMP VALUE.
1530 20   ICOMP = IVAL(FP)
1540      GO TO 70
1550* - - SET IDRFC VALUE.
1560 25   IDRFC = IVAL(FP)
1570      GO TO 70
1580* - - SET CHARACTER TARGET VALUE.
1590*     CHECK FOR DRFC = MC.
1600 30   IF(IDRFC.NE.4) GO TO 32
1610      TARGET = MFGCOD(IVAL(FP))
1620      GO TO 70
1630* - - CHECK FOR DRFC = TX.
1640 32   IF(IDRFC.NE.15) GO TO 34
1650      TEXT = CTEXT(IVAL(FP))
1660      GO TO 70
1670 34   TARGET = CVAL(FP)
1680      GO TO 70
1690* - - SET TARGET VALUE FROM OTHER VALUES IN RECORD.
1700 35   ITAR = IVAL(FP)
1710      CALL CONCAT(TARGET, 1, REC, FRCB(ITAR), NCBR(ITAR))
1720      GO TO 70
1730* - - SET TARGET 2 VALUES.
1740 40   TAR2 = CVAL(FP)
1750      GO TO 70
1760* - - SET END OF TABLE (EOT) FLAG.
1770 45   EOT = .T.
1780      GO TO 75
1790C
1800 70   IF(FUNC(FP).GT.0) GO TO 5
1810* - - UPDATE COMPLETE.
1820      IF(SKIP) GO TO 130
1830C
1840 75   IF(MAJF) GO TO 135
1850      IF(.NOT. EOT) GO TO 78
1860      KEYL = 0
1870      MP = MPL(FP-1)
1880      B = 0
1890      IF(MP.EQ.1) GO TO 144
1900      DO 76 J = 1, MP
1910      IF(LEVEL(J).EQ.0) GO TO 77
1920 76   CONTINUE
1930 77   B = MP - J
1940      GO TO 144
1950C
1960 78   IF(IDRFC.EQ.15) GO TO 80
1970      K = KOMPCH(TARGET, 1, REC, FRCB(IDRFC), NCBR(IDRFC))
1980      GO TO 90
```

```
1990*  - - COMPARE TEXT STRINGS.
2000 90   K = KOMPCH (TEXT, 1, REC, FRCB(IDRFC), NCBR(IDRFC))
2010 90   GO TO (102, 104, 106, 108, 110, 112), ICOMP
2020*  - - EQ CHECK.
2030 102  IF(K.NE.0) GO TO 200
2040      GO TO 120
2050*  - - NE CHECK.
2060 104  IF(K.EQ.0) GO TO 200
2070      GO TO 120
2080*  - - GE CHECK.
2090 106  IF(K.NE.0) GO TO 110
2100      GO TO 120
2110*  - - LE CHECK.
2120 108  IF(K.NE.0) GO TO 112
2130      GO TO 120
2140*  - - GT CHECK.
2150 110  IF(K.NE.1) GO TO 200
2160      GO TO 120
2170*  - - LT CHECK.
2180 112  IF(K.NE.-1) GO TO 200
2190*  - - HAS PASSED THIS CHECK...
2200*     SET OUTCOM TO TRUE.
2210 120  OUTCOM(MPL(FP)) = 1
2220*  - - SKIP THE REST OF THE CHECKS FOR THIS LEVEL.
2230      SKIP = .T.
2240*  - - SAVE LAST MP VALUE.
2250      LMP = MPL(FP)
2260      GO TO 5
2270C
2280*  - - END OF THIS FUNC.  CHECK FOR CHANGE IN LEVEL.
2290 130  IF(MPL(FP).LE.LMP) GO TO 5
2300*  - - END OF OLD LEVEL.  BEGIN PROCESSING NEW LEVEL.
2310      SKIP = .F.
2320      GO TO 75
2330C
2340*  - - EVALUATE PRECEEDING.
2350 135  MP = MPL(FP)
2360      B = 1
2370      KEYL = LEVEL(MP - 1)
2380      DO 140 B = 2, MP - 1
2390      IF(LEVEL(MP-B).LT.KEYL) GO TO 142
2400      IF(B.EQ.MP-1) GO TO 144
2410 140  CONTINUE
2420 142  B = B - 1
2430 144  SUMER = 0
2440      SUMIT = .F.
2450      K = 0
2460 146  IF(LOP(MP-B).EQ.1) SUMIT = .T.
2470 147  IF(LEVEL(MP-B).NE.KEYL) GO TO 150
2480      SUMER = SUMER + OUTCOM(MP-B)
2490      K = K + 1
2500 150  B = B - 1
```

```
2510      IF(B.GT.0) GO TO 147
2520      IF(.NOT. EOT) GO TO 152
2530      SUMER = SUMER + OUTCOM(MP)
2540      K = K + 1
2550* - - EVALUATE KEYL.
2560 152  IF(SUMIT) GO TO 155
2570* - - EVALUATING FOR 'OR' EXPRESSION.
2580      IF(SUMER.GT.0) GO TO 165
2590* - - OUTCOM IS FALSE.
2600      GO TO 170
2610* - - EVALUATING FOR 'AND' EXPRESSION.
2620 155  IF(SUMER.EQ.K) GO TO 165
2630* - - OUTCOM IS FALSE.
2640      GO TO 170
2650* - - MARK OUTCOM TRUE.
2660 165  OUTCOM(MP) = 1
2670      IF(EOT) GO TO 175
2680* - - CHECK FOR EOT.
2690 170  MAJF = .F.
2700      IF(.NOT. EOT) GO TO 5
2710* - - DESELECT THIS RECORD.
2720**************************************************************
2730      ISTAT = 1
2740      GO TO 300
2750**************************************************************
2760C
2770**************************************************************
2780* - - SELECT THIS RECORD.
2790 175  ISTAT = 2
2800      GO TO 300
2810**************************************************************
2820C
2830* - - FAILED LAST TEST.
2840*     IF THERE IS STILL AN 'OR' PROCESS TO BE DONE, PRESS ON.
2850 200  IF(MINF.EQ.2) GO TO 5
2860* - - IF THERE IS STILL A 'TO' PROCESS IN WORK, PRESS ON.
2870      IF(MINF.EQ.3) GO TO 210
2880* - - HAS FAILED AND NO CHANCE TO REDEEM IT.
2890*     OUTCOM STAYS 0 (FALSE).
2900* - - PRESS ON WITH PROCESSING.
2910      GO TO 5
2920* - - CHECK OUT 'TO' FUNCTION.
2930 210  INCR = .F.
2940      L = KOMPCH (TAR2, 1, TARGET, 1, 6)
2950      IF(L.LT.0) GO TO 215
2960* - - RANGE IS DECREASING.
2970      IF(K.EQ.-1) GO TO 220
2980* - - TEST VALUE CAN'T BE WITHIN RANGE.  OUTCOM STAYS FALSE.
2990      GO TO 5
3000* - - RANGE IS INCREASING.
3010 215  INCR = .T.
3020      IF(K.EQ.1) GO TO 220
```

```
3030* - - TEST VALUE CAN'T BE WITHIN RANGE.  OUTCOM STAYS FALSE.
3040      GO TO 5
3050* - - CHECK VALUE AGAINST TARGET.
3060 220  K = KOMPCH (TAR2, 1, REC, FRCB(IDRFC), NCBR(IDRFC))
3070      IF(INCR) GO TO 225
3080      IF(K.GE.0) GO TO 230
3090* - - NOT IN RANGE.  OUTCOM IS FALSE.
3100      GO TO 5
3110C
3120 225  IF(K.LE.0) GO TO 230
3130* - - NOT IN RANGE.  OUTCOM IS FALSE.
3140      GO TO 5
3150* - - OUTCOM IS TRUE.  VALUE IS IN RANGE.
3160 230  OUTCOM(NPL(FP)) = 1
3170      GO TO 5
3180C
3190 300  RETURN
3200      END
```

```
1000CHEADER   LIST HEADER SECTIONS (ID, INFO, DR)/UPDATE INFO SEC.
1010C
1020      SUBROUTINE HEADER
1030C
1040      CHARACTER  CPUC*2(15), CPUN*8(15), LANGC*1(10), LANGN*8(10),
1050&                SIMCOD*1(20), SIMNAM*8(20), CARD(450), REC*42(10),
1060&                FNAME*6, DATES*8(3), CINFO*450, CREC*90(5), ANS*1,
1070&                BLANKS*43/' '/, OUT*48, FORM1*8/'(1X,A  )'/,
1080&                MCOD*1, DCPU*2
1090C
1100      INTEGER  SECNO, CONT, CPT, DIR(3), INFO(3), DATAS(3), RNUM,
1110&              ISEC, INSTR, IDATA, SIMTXT(20)
1120C
1130      REAL  CPUS(15), CPUT(15), LANGS(10), SIMFAC(20), TIMING
1140C
1150      LOGICAL  DIREUP, INFOUP, DATAUP, WINFO, CPUD, LANGD,
1160&              SIMD, DONE
1170C
1180      EQUIVALENCE  (CARD, REC(1)), (CINFO, CREC(1))
1190C
1200      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
1210C
1220      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
1230C
1240      COMMON /INSEC/ CPUC, CPUN, CPUS, CPUT, LANGC, LANGN, LANGS
1250C
1260      COMMON /INFOS/ SIMCOD, SIMNAM, SIMFAC, SIMTXT
1270C
1280      SIMD = .F.; DONE = .F.
1290 10   PRINT, "ENTER: L (LIST), U (UPDATE), OR E (END)?"
1300      READ, ANS
1310      IF(ANS.EQ.'E') GO TO 800
1320      IF(ANS.EQ.'U') GO TO 300
1330      IF(ANS.NE.'L') GO TO 10
1340* - - WE ARE IN LIST MODE.
1350 25   PRINT, "ENTER: I (ID), C (CPU/LANG), S (SIM INFO), ",
1360&           "T (SIM TEXT), E (END)."
1370      READ, ANS
1380      IF(ANS.EQ.'E') GO TO 10
1390      IF(ANS.EQ.'I') CALL !DLIST
1400      IF(ANS.EQ.'S') GO TO 85
1410      IF(ANS.EQ.'T') GO TO 140
1420      IF(ANS.NE.'C') GO TO 25
1430* - - LIST INFO DATA (CPU AND LANG).
1440      PRINT 35
1450 35   FORMAT (1H0, "--------CPU INFO---------   ----LANGUAGE INFO",
1460&           "---", /, "CPU", 11X, "BYTES/ TIME   LANG LANGUAGE  ",
```

```
1470&           "CONV",/, "CODE CPU NAME  WORD  CONV   CODE   NAME   ",
1480&           "FACTOR", //)
1490      DO 80 I = 1, 15
1500      CPUD = .F.; LANGD = .F.
1510      OUT = BLANKS
1520      IF(CPUC(I).EQ.'  ') GO TO 50
1530      CPUD = .T.
1540      ENCODE (OUT, 40) CPUC(I), CPUN(I), CPUS(I), CPUT(I)
1550 40   FORMAT (1X, A2, 2X, A8, F6.1, F6.2)
1560 50   IF(I.GT.10) GO TO 60
1570      IF(LANGC(I).EQ.' ') GO TO 60
1580      LANGD = .T.
1590      ENCODE (OUT, 55) LANGC(I), LANGN(I), LANGS(I)
1600 55   FORMAT (T31, A1, 2X, A8, F6.2)
1610 60   IF(CPUD .OR. LANGD) GO TO 65
1620* - - SKIP TO END OF LOOP.
1630      GO TO 80
1640* - - PRINT 'OUT'.
1650 65   NCH = 48
1660      IF(.NOT. LANGD) NCH = 26
1670      ENCODE (FORM1, 70) NCH
1680 70   FORMAT (T6, I2)
1690      PRINT FORM1, OUT
1700 80   CONTINUE
1710      GO TO 25
1720C
1730* - - LIST SIMULATOR INFO.
1740 85   PRINT 90
1750 90   FORMAT (1H0, " SIM   SIMULATOR  MISS  CMPLXT  INFO ",
1760&           "TEXT  INSTRUCTIONS     DATA", 7X, "TIMING", /,
1770&           " CODE    NAME     CODE  FACTOR  SECTOR NO   ",
1780&           "(IN BYTES)   (IN BYTES)  (MSEC/SEC)", //)
1790      J = 0
1800      SECNO = INFO(1) + 1
1810 95   CALL RSEC (CARD, SECNO, CONT, $900)
1820      DO 120 I = 1, 10
1830      J = J + 1
1840      IF(SIMCOD(J).EQ.' ') GO TO 120
1850      DECODE (REC(I), 100) MCOD, INSTR, IDATA, TIMING
1860 100  FORMAT (T10, A1, T19, 2I7, F8.2)
1870      PRINT 110, SIMCOD(J), SIMNAM(J), MCOD, SIMFAC(J), SIMTXT(J),
1880&               INSTR, IDATA, TIMING
1890 110  FORMAT (3X, A1, 4X, A8, 5X, A1, F8.1, I9, 1X, 2(5X, I8),
1900&           4X, F8.2)
1910 120  CONTINUE
1920      IF(J.EQ.20) GO TO 130
1930      SECNO = CONT
1940      GO TO 95
1950C
1960 130  PRINT, "DO YOU WANT TO LIST INFORMATION TEXT (Y OR N)?"
1970      READ, ANS
1980      IF(ANS.EQ.'N') GO TO 10
```

```
1990* - - WANTS TO LIST INFO TEXT.
2000 140  PRINT, "ENTER THE 1 CHARACTER SIM CODE."
2010 142  READ, ANS
2020      DO 150 I = 1, 20
2030      IF(ANS.EQ.SIMCOD(I)) GO TO 170
2040 150  CONTINUE
2050      PRINT 160, ANS
2060 160  FORMAT (1X, "THERE IS NO TEXT FOR SIM CODE ", A1)
2070      GO TO 130
2080 170  SECNO = SIMTXT(I)
2090      IF(SIMD) GO TO 404
2100      IF(SECNO.GT.0) GO TO 190
2110      PRINT 160, ANS
2120      GO TO 130
2130* - - READ INFO TEXT AND PRINT IT OUT.
2140 190  CALL RSEC (CINFO, SECNO, CONT, $900)
2150      DO 200 K = 1, 5
2160      PRINT 195, CREC(K)
2170 195  FORMAT (1X, A80)
2180 200  CONTINUE
2190      IF(CONT.EQ.0) GO TO 130
2200      SECNO = CONT
2210      GO TO 190
2220C
2230* - - ADD/CHANGE/DELETE INFO ENTRIES.
2240 300  PRINT, "DO YOU WANT TO ADD A (ADD), C (CHANGE), D (DELETE), ",
2250&           "L (LIST), OR E (END)?"
2260      READ, ANS
2270      IF(ANS.EQ.'E') GO TO 800
2280      IF(ANS.EQ.'D') GO TO 410
2290      IF(ANS.EQ.'A') GO TO 310
2300      IF(ANS.EQ.'C') GO TO 500
2310      IF(ANS.EQ.'L') GO TO 25
2320      GO TO 300
2330* - - ADD NEW ENTRIES.
2340 310  PRINT, "DO YOU WANT TO ADD TO C (CPU), L (LANGUAGE), OR ",
2350&           "S (SIMULATOR) INFO?"
2360      READ, ANS
2370      IF(ANS.EQ.'C') GO TO 320
2380      IF(ANS.EQ.'L') GO TO 340
2390      IF(ANS.EQ.'S') GO TO 380
2400      GO TO 300
2410C
2420* - - ADD TO CPU INFO.
2430*     FIND FIRST BLANK CPU SLOT.
2440 320  DO 330 I = 1, 15
2450      IF(CPUC(I).EQ.'  ') GO TO 335
2460 330  CONTINUE
2470      PRINT, "NO MORE ROOM IN CPU INFO TABLE."
2480      GO TO 300
2490 335  PRINT, "ENTER 2 CHARACTER CPU CODE."
2500      READ, CPUC(I)
```

```
      PRINT, "ENTER 8 CHARACTER CPU NAME."
      READ, CPUN(I)
      PRINT, "ENTER NUMBER OF BYTES/WORD IN X.X FORMAT."
      READ, CPUS(I)
      PRINT, "ENTER THE TIME CONVERSION FACTOR IN X.XX FORMAT."
      READ, CPUT(I)
      INFOUP = .T.
      GO TO 416
C
* - - ADD TO LANGUAGE INFO.
*     FIND FIRST BLANK LANGUAGE SLOT.
  340 DO 350 I = 1, 10
      IF(LANGC(I).EQ.' ') GO TO 360
  350 CONTINUE
      PRINT, "NO MORE ROOM IN LANGUAGE INFO TABLE."
      GO TO 300
  360 PRINT, "ENTER 1 CHARACTER LANGUAGE CODE."
      READ, LANGC(I)
      PRINT, "ENTER 8 CHARACTER LANGUAGE NAME."
      READ, LANGN(I)
      PRINT, "ENTER CONVERSION FACTOR IN XX.XX FORMAT."
      READ, LANGS(I)
      INFOUP = .T.
      GO TO 416
C
  380 PRINT, "DO YOU WANT TO CREATE A NEW SIMULATOR ENTRY 'N' ",
     &       "OR JUST ADD"
      PRINT, "TEXT TO AN EXISTING ENTRY 'T'?"
      READ, ANS
      IF(ANS.EQ.'N') GO TO 385
      IF(ANS.EQ.'T') GO TO 402
      GO TO 380
* - - ADD TO SIMULATOR INFO.
*     FIND FIRST BLANK SIMULATOR SLOT.
  385 DO 390 I = 1, 20
      IF(SIMCOD(I).EQ.' ') GO TO 400
  390 CONTINUE
      PRINT, "NO MORE ROOM IN SIMULATOR INFO TABLE."
      GO TO 300
  400 PRINT, "ENTER 1 CHARACTER SIMULATOR CODE."
      READ, SIMCOD(I)
      PRINT, "ENTER 8 CHARACTER SIMULATOR NAME."
      READ, SIMNAM(I)
      PRINT, "ENTER COMPLEXITY FACTOR IN XX.X FORMAT."
      READ, SIMFAC(I)
      INFO(3) = INFO(3) + 1
      GO TO 406
  402 PRINT, "ENTER 1 CHARACTER SIM CODE FOR ENTRY YOU WANT TO ",
     &       "ADD TEXT TO."
      SIMD = .T.; GO TO 142
  404 SIMD = .F.
      IF(SECNO.EQ.0) GO TO 406
```

```
3030* - - THERE ALREADY IS A TEXT SECTION FOR THIS ENTRY.
3040      PRINT, "THIS SIM CODE ALREADY HAS AN INFO TEXT SECTION ",
3050&           "ESTABLISHED."
3060      PRINT, "DO YOU WANT TO ADD MORE TEXT (Y OR N)?"
3070      READ, ANS
3080      IF(ANS.EQ.'N') GO TO 300
3090* - - FIND END OF EXISTING TEXT.
3100 4042 CALL RSEC (CINFO, SECNO, CONT, $900)
3110      IF(CONT.EQ.0) GO TO 4043
3120      SECNO = CONT
3130      GO TO 4042
3140 4043 DO 405 J = 1, 5
3150      JI = 6 - J
3160      DO 4044 K = 1, 80
3170      L = K
3180      CALL CONCAT (MCOD, 1, CREC(JI), L, 1)
3190      IF(MCOD.NE.' ') GO TO 4052
3200 4044 CONTINUE
3210 405  CONTINUE
3220* - - IF JI EQ 5, THIS SECTOR IS FULL.
3230 4052 IF(JI.EQ.5) SIMD = .T.
3240      GO TO 409
3250* - - OBTAIN NEXT SECTOR NUMBER.
3260 406  SECNO = DATAS(3)
3270      SINTXT(I) = SECNO
3280      CONT = SECNO
3290      DATAS(3) = DATAS(3) + 1
3300      JI = 0
3310      IF(DATAS(2).GT.(DATAS(3) - DATAS(1))) GO TO 409
3320 407  PRINT 408, DATAS(3)
3330 408  FORMAT (1X, "HAVE RUN OUT OF ROOM FOR TEXT AT SECTOR",
3340&           " NUMBER ", I4)
3350      GO TO 300
3360 409  PRINT, "ENTER YOUR TEXT, ONE LINE AT A TIME WITH A ",
3370&           "MAXIMUM OF 80 CHARACTERS"
3380      PRINT, "PER LINE.  ENTER 'ZZZ' TO SIGNAL 'NO MORE TEXT ",
3390&           "TO ENTER'."
3400      IF(SIMD) GO TO 4112
3410      IF(JI.EQ.1) GO TO 4092
3420      JI = JI + 1
3430 4092 DO 4093 L = JI, 5
3440      CREC(L) = ' '
3450 4093 CONTINUE
3460 410  READ 411, CREC(JI)
3470 411  FORMAT (A80)
3480      IF(CREC(JI).EQ.'ZZZ') GO TO 412
3490      JI = JI + 1
3500      IF(JI.LT.6) GO TO 410
3510      IF(CONT.EQ.0) GO TO 4112
3520      SECNO = CONT
3530 4112 CONT = DATAS(3)
3540      SIMD = .F.
```

```
3550      DATAS(3) = DATAS(3) + 1
3560      IF(DATAS(2).LE.(DATAS(3) - DATAS(1))) CONT = 0
3570      GO TO 413
3580 412  IF(JI.EQ.1) GO TO 416
3590* - -  WIPE OUT 'ZZZ'.
3600      CREC(JI) = ' '
3610      IF(CONT.EQ.0) GO TO 4122
3620      SECNO = CONT
3630      CONT = 0
3640 4122 DONE = .T.
3650 413  CALL WSEC (CINFO, SECNO, CONT)
3660      INFOUP = .T.
3670      IF(DONE) GO TO 416
3680      IF(CONT.EQ.0) GO TO 407
3690      JI = 1
3700      GO TO 4092
3710 416  PRINT, "ADD FUNCTION COMPLETE."
3720      DONE = .F.
3730      GO TO 300
3740C
3750* - -  DELETE INFO.
3760 418  PRINT, "DO YOU WANT TO DELETE C (CPU), L (LANGUAGE), OR ",
3770&           "S (SIMULATOR) INFO?"
3780      READ, ANS
3790      IF(ANS.EQ.'C') GO TO 420
3800      IF(ANS.EQ.'L') GO TO 440
3810      IF(ANS.EQ.'S') GO TO 460
3820      GO TO 300
3830* - -  DELETE CPU INFO.
3840 420  PRINT, "ENTER THE 2 CHARACTER CPU CODE OF ENTRY TO BE DELETED."
3850      READ, DCPU
3860      DO 430 I = 1, 15
3870      IF(CPUC(I).NE.DCPU) GO TO 430
3880      CPUC(I) = '  '
3890      GO TO 480
3900 430  CONTINUE
3910      PRINT, "NO ENTRY FOR CPU CODE ", DCPU
3920      GO TO 300
3930C
3940* - -  DELETE LANGUAGE INFO.
3950 440  PRINT, "ENTER 1 CHARACTER LANGUAGE CODE OF ENTRY TO BE DELETED."
3960      READ, MCOD
3970      DO 450 I = 1, 10
3980      IF(LANGC(I).NE.MCOD) GO TO 450
3990      LANGC(I) = ' '
4000      GO TO 480
4010 450  CONTINUE
4020      PRINT, "NO ENTRY FOR LANGUAGE CODE ", MCOD
4030      GO TO 300
4040C
4050* - -  DELETE SIMULATOR ENTRY.
4060 460  PRINT, "ENTER THE 1 CHARACTER SIM CODE OF ENTRY TO BE DELETED."
```

```
      READ, MCOD
      DO 470 I = 1, 20
      IF(SIMCOD(I).NE.MCOD) GO TO 470
      SIMCOD(I) = ' '
      GO TO 480
470   CONTINUE
      PRINT, "NO ENTRY FOR SIMULATOR CODE ", MCOD
      GO TO 300
480   PRINT, "DELETE FUNCTION COMPLETE."
      INFOUP = .T.
      GO TO 300
C
C - - CHANGE INFO DATA.
500   PRINT, "DO YOU WANT TO CHANGE C (CPU), L (LANGUAGE), OR ",
     &       "S (SIMULATOR) INFO?"
      READ, ANS
      IF(ANS.EQ.'C') GO TO 510
      IF(ANS.EQ.'L') GO TO 580
      IF(ANS.EQ.'S') GO TO 640
      GO TO 300
C - - CHANGE CPU INFO.
510   PRINT, "ENTER 2 CHARACTER CPU CODE OF ENTRY TO BE CHANGED."
      READ, DCPU
      DO 520 I = 1, 15
      IF(CPUC(I).EQ.DCPU) GO TO 530
520   CONTINUE
      PRINT, "NO ENTRY FOR CPU CODE ", DCPU
      GO TO 300
530   PRINT, "DO YOU WANT TO CHANGE CPU CODE '1', CPU NAME '2', ",
     &       "BYTES/WORD '3', OR TIME CONV'4'?"
      READ, NUMB
      IF(NUMB.LT.1 .OR. NUMB.GT.4) GO TO 530
      GO TO (540, 550, 560, 565), NUMB
540   PRINT, "ENTER NEW CPU CODE."
      READ, CPUC(I)
      GO TO 570
550   PRINT, "ENTER NEW CPU NAME (8 CHARACTERS MAX)."
      READ, CPUN(I)
      GO TO 570
560   PRINT, "ENTER NEW BYTES/WORD VALUE IN X.X FORMAT."
      READ, CPUS(I)
      GO TO 570
565   PRINT, "ENTER NEW TIME CONVERSION FACTOR IN X.XX FORMAT."
      READ, CPUT(I)
570   PRINT, "CHANGE COMPLETE."
      INFOUP = .T.
      GO TO 300
C
C - - CHANGE LANGUAGE INFO.
580   PRINT, "ENTER THE 1 CHARACTER LANGUAGE CODE FOR ENTRY TO BE",
     &       " CHANGED."
      READ, MCOD
```

```
4590      DO 590 I = 1, 10
4600      IF(LANGC(I).EQ.MCOD) GO TO 595
4610 590  CONTINUE
4620      PRINT, "NO ENTRY FOR LANGUAGE CODE ", MCOD
4630      GO TO 300
4640 595  PRINT, "DO YOU WANT TO CHANGE LANGUAGE CODE '1', LANG NAME '2'",
4650&           " OR CONV FACTOR '3'?"
4660      READ, NUMB
4670      IF(NUMB.LT.1 .OR. NUMB.GT.3) GO TO 580
4680      GO TO (600, 610, 620), NUMB
4690 600  PRINT, "ENTER NEW LANGUAGE CODE."
4700      READ, LANGC(I)
4710      GO TO 570
4720 610  PRINT, "ENTER NEW LANGUAGE NAME (8 CHARACTERS MAX)."
4730      READ, LANGN(I)
4740      GO TO 570
4750 620  PRINT, "ENTER NEW CONVERSION FACTOR IN XX.XX FORMAT."
4760      READ, LANGS(I)
4770      GO TO 570
4780C
4790 640  PRINT, "ENTER THE 1 CHARACTER SIMULATOR CODE OF ENTRY TO BE ",
4800&           "CHANGED."
4810      READ, MCOD
4820      DO 650 I = 1, 20
4830      IF(SIMCOD(I).EQ.MCOD) GO TO 660
4840 650  CONTINUE
4850 652  PRINT, "NO ENTRY FOR SIMCOD ", MCOD
4860      GO TO 300
4870 660  PRINT, "DO YOU WANT TO CHANGE SIM CODE '1', SIM NAME '2', ",
4880&           "CMPLXT FACT '3', OR TEXT '4'?"
4890      READ, NUMB
4900      IF(NUMB.LT.1 .OR. NUMB.GT.4) GO TO 660
4910      GO TO (670, 680, 690, 700), NUMB
4920 670  PRINT, "ENTER NEW SIMULATOR CODE."
4930      READ, SIMCOD(I)
4940      GO TO 570
4950 680  PRINT, "ENTER NEW SIMULATOR NAME."
4960      READ, SIMNAM(I)
4970      GO TO 570
4980 690  PRINT, "ENTER NEW COMPLEXITY FACTOR IN XX.X FORMAT."
4990      READ, SIMFAC(I)
5000      GO TO 570
5010 700  IF(SIMTXT(I).GT.0) GO TO 706
5020      PRINT 702, MCOD
5030 702  FORMAT (1X, "THERE IS NO TEXT FOR SIM CODE ", A1,
5040&           ".  USE ADD FUNCTION TO INSERT TEXT.")
5050      GO TO 300
5060 706  PRINT, "LINES ARE NUMBERED SEQUENTIALLY STARTING WITH THE ",
5070&           "FIRST LINE OF TEXT, i.e.,"
5080      PRINT, "1, 2, 3, 4, etc."
5090      LSEC = 0
5100 708  PRINT, "ENTER THE NUMBER OF THE LINE OF TEXT TO BE CHANGED."
```

```
5110 710  READ, NUMB
5120* - - COMPUTE SECTOR.
5130      IR = MOD(NUMB, 5)
5140      ISEC = NUMB / 5
5150      IF(IR.GT.0) ISEC = ISEC + 1
5160      IF(IR.EQ.0) IR = 5
5170* - - IF IN SAME SECTOR AS LAST CHANCE, SKIP WRITE/READ.
5180      IF(ISEC.EQ.LSEC) GO TO 740
5190* - - IF LSEC NE 0 WRITE THE CURRENT SECTOR BACK TO THE FILE.
5200      IF(LSEC.EQ.0) GO TO 720
5210* - - WRITE SECTOR BACK TO FILE.
5220      CALL WSEC (CINFO, SECNO, CONT)
5230      INFOUP = .T.
5240      IF(ISEC.GT.LSEC)GO TO 721
5250* - - READ FIRST SECTOR.
5260 720  SECNO = SIMTXT(I)
5270      LSEC = 0
5280      GO TO 722
5290* - - READ NEXT SECTOR.
5300 721  SECNO = CONT
5310 722  CALL RSEC (CINFO, SECNO, CONT, $900)
5320      LSEC = LSEC + 1
5330      IF(LSEC.EQ.ISEC) GO TO 740
5340      IF(CONT.EQ.0) GO TO 725
5350      GO TO 720
5360 725  PRINT 730, LSEC * 5
5370 730  FORMAT (1X, "THERE ARE ONLY ", I2, " LINES OF TEXT FOR ",
5380&           "THIS SIM CODE.")
5390      GO TO 700
5400 740  PRINT 750, NUMB
5410 750  FORMAT (1X, "ENTER THE NEW TEXT TO REPLACE LINE ", I2,
5420&           " (A MAXIMUM OF 80 CHARACTERS).")
5430      READ 411, CREC(IR)
5440      PRINT, "DO YOU WANT TO CHANGE ANOTHER LINE (Y OR N)?"
5450      READ, ANS
5460      IF(ANS.EQ.'Y') GO TO 700
5470* - - WRITE SECTOR BACK TO FILE.
5480      CALL WSEC (CINFO, SECNO, CONT)
5490      GO TO 570
5500C
5510* - - WRITE INFO SECTIONS BACK TO DATA BASE.
5520 800  IF(.NOT. INFOUP) RETURN
5530      CALL RWHEAD (2)
5540      INFOUP = .T.
5550      RETURN
5560 900  PRINT, "EOF ON DB READ DURING HEADER LIST/UPDATE."
5570      RETURN
5580      END
```

```
      SUBROUTINE IDLIST
C
C - - ROUTINE TO PERFORM FORMATED DUMP OF DATA BASE ID SECTION.
C
      CHARACTER  FNAME*6, DATES*8(3)
C
      INTEGER  DIR(3), INFO(3), DATAS(3), SECNO, RNUM
C
      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
C
C*********************************************************************
C
      WRITE (06, 100)
100   FORMAT (1H0, 8X, "SIMSIZ DATA BASE IDENTIFICATION SECTION",//)
      WRITE (06, 110) FNAME
110   FORMAT (4X, "FILE NAME: ", A6, //)
      WRITE (06, 120) DATES(1), DATES(2), DATES(3)
120   FORMAT (4X, "DATES OF LAST CHANGES TO:", /, 8X,
     &           "DIRECTORY - - ", A8, /, 8X,
     &           "INFORMATION - ", A8, /, 8X,
     &           "DATA  - - - - ", A8, //)
      WRITE (06, 130) LSIZE, RNUM
130   FORMAT (4X, "FILE SIZE:", I4, " LLINKS",
     &         //, 4X, "NUMBER OF DATA RECORDS:", I5,//)
      WRITE (06, 140) (DIR(I), I = 1, 3), (INFO(I), I = 1, 3),
     &                (DATAS(I), I = 1, 3)
140   FORMAT (16X, "STARTING   SECTORS", /, 17X,
     &           "SECTOR   ALLOCATED", /, 4X, "DIRECTORY - - ",
     &           I4, 6X, I4, 5X, I4, " CURRENT ENTRIES", //, 4X,
     &           "INFORMATION - ", I4, 6X, I4, 5X, I4,
     &           " RECORDS IN USE",
     &           //, 4X, "DATA  - - - - ", I4, 6X, I4, 5X, I4,
     &           " NEXT AVAILABLE")
      RETURN
      END
```

SUBROUTINE IOSET

```
CRWHEAD  SUB TO READ AND WRITE ID, INFO, AND DR SECTIONS.
      SUBROUTINE RWHEAD(MODE)
C
C     IF MODE = 1   READS DATA BASE AND LOADS COMMONS
C                   IDSEC, INSEC, INFOS, AND DRSEC.
C     IF MODE = 2   PUTS CONTENTS OF COMMONS IDSEC, INSEC,
C                   INFOS, AND DRSEC INTO DATA BASE.
C
      CHARACTER  FNAME*6, DATES*8(3), CARD*450, REC*30(15),
     &             TODAY*8, SYSCOD*2(40), SYSNAM*12(40),
     &             IREC*18(25), FREC*42(10), CPUC*2(15), CPUN*8(15),
     &             LANGC*1(10), LANGN*8(10), SIMCOD*1(20),
     &             SIMNAM*8(20)
C
      INTEGER  SECNO, LSIZE, DIR(3), INFO(3), DATAS(3),
     &           CONT, DSSEC(40), DESEC(40), DNSEC(40),
     &           NDEL(40), RNUM, SIMTXT(20)
C
      REAL  CPUS(15), CPUT(15), LANGS(10), SIMFAC(20)
C
      LOGICAL  DIREUP, INFOUP, DATAUP, WINFO
C
      EQUIVALENCE  (CARD, REC(1), IREC(1), FREC(1))
C
      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
C
      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
C
      COMMON /INSEC/ CPUC, CPUN, CPUS, CPUT, LANGC, LANGN, LANGS
C
      COMMON /INFOS/ SIMCOD, SIMNAM, SIMFAC, SIMTXT
C
      COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DNSEC, NDEL
C
*************************************************************************
C
* - - CHECK MODE TO SEE IF WE ARE READING OR WRITING DATA BASE.
      IF(MODE.EQ.2) GO TO 200
* - - WE ARE IN READ MODE.
*     RESET UPDATE FLAGS.
      DIREUP = .F.
      INFOUP = .F.
      DATAUP = .F.
      CALL RSEC (CARD, 1, CONT, $900)
      DECODE (CARD, 50) FNAME, DATES(1), DATES(2), DATES(3),
     &LSIZE, (DIR(I), INFO(I), DATAS(I), I = 1,3), RNUM
 50   FORMAT (A6, 3A8, 11I4)
```

```
1470      J = 0
1480      K = 15
1490      SECNO = DIR(1)
1500      IF(DIR(3).LT.15) K = DIR(3)
1510 60   CALL RSEC (CARD, SECNO, CONT, $900)
1520      DO 100 I = 1, K
1530      J = J + 1
1540      DECODE (REC(I), 70) SYSCOD(J), SYSNAM(J), DSSEC(J),
1550&                 DESEC(J), DNSEC(J), NDEL(J)
1560 70   FORMAT (A2, A12, 2I4, 2I3, 2X)
1570 100  CONTINUE
1580      IF(DIR(3).EQ.J) GO TO 110
1590      SECNO = CONT
1600      K = DIR(3) - J
1610      IF(K.GT.15) K = 15
1620      GO TO 60
1630C
1640* - - READ INFO SECTION.
1650 110  SECNO = INFO(1)
1660      CALL RSEC (CARD, SECNO, CONT, $900)
1670      J = 0
1680      DO 140 I = 1, 25
1690      IF(I.GT.15) GO TO 125
1700      DECODE (IREC(I), 120) CPUC(I), CPUN(I), CPUS(I), CPUT(I)
1710 120  FORMAT (A2, A8, F3.1, F4.2)
1720      GO TO 140
1730 125  J = J + 1
1740      DECODE (IREC(I), 130) LANGC(J), LANGN(J), LANGS(J)
1750 130  FORMAT (A1, A8, F5.2)
1760 140  CONTINUE
1770      SECNO = CONT
1780      J = 0
1790 145  CALL RSEC (CARD, SECNO, CONT, $900)
1800      DO 160 I = 1, 10
1810      J = J + 1
1820      DECODE (FREC(I), 150) SIMCOD(J), SIMNAM(J), SIMFAC(J),
1830&                           SIMTXT(J)
1840 150  FORMAT (A1, A8, T11, F4.1, I4)
1850 160  CONTINUE
1860* - - CHECK TO SEE IF ALL 20 RECORDS HAVE BEEN LOADED.
1870      IF(J.EQ.20) GO TO 170
1880      SECNO = CONT
1890      GO TO 145
1900 170  RETURN
1910C
1920**************************************************************
1930C
1940* - - WRITE TO DATA BASE.
1950*     UPDATE DATES IN ID SECTION AS NECESSARY.
1960 200  CALL DATIM (TODAY, TIME)
1970      IF(DIREUP) DATES(1) = TODAY
1980      IF(INFOUP) DATES(2) = TODAY
```

```
1990      IF(DATAUP) DATES(3) = TODAY
2000      ICNT = 2
2010      ENCODE (CARD, 50) FNAME, DATES(1), DATES(2), DATES(3),
2020&     LSIZE, (DIR(I),INFO(I),DATAS(I), I = 1, 3), RNUM
2030      CALL WSEC (CARD, 1, 2)
2040      J = 0
2050      K = 15
2060      SECNO = DIR(1)
2070      IF(DIR(3).LT.15) K = DIR(3)
2080 205  CALL RSEC (CARD, SECNO, CONT, $900)
2090      DO 210 I = 1, K
2100      J = J + 1
2110      ENCODE (REC(I), 70) SYSCOD(J), SYSNAM(J), DSSEC(J),
2120&                         DESEC(J), DNSEC(J), NDEL(J)
2130 210  CONTINUE
2140      ICNT = ICNT + 1
2150      CONT = ICNT
2160      CALL WSEC (CARD, SECNO, CONT)
2170      IF(DIR(3).EQ.J) GO TO 220
2180      SECNO = CONT
2190      K = DIR(3) - J
2200      IF(K.GT.15) K = 15
2210      GO TO 205
2220C
2230* - - WRITE INFO SECTION.
2240 220  J = 0
2250      DO 240 I = 1, 25
2260      IF(I.GT.15) GO TO 230
2270      ENCODE (IREC(I),120) CPUC(I), CPUN(I), CPUS(I), CPUT(I)
2280      GO TO 240
2290 230  J = J + 1
2300      ENCODE (IREC(I), 130) LANGC(J), LANGN(J), LANGS(J)
2310 240  CONTINUE
2320      SECNO = INFO(1)
2330      CONT = SECNO + 1
2340      CALL WSEC (CARD, SECNO, CONT)
2350      J = 0
2360* - - READ OLD SIM INFO SECTOR.
2370 245  SECNO = CONT
2380      CALL RSEC (CARD, SECNO, CONT, $900)
2390      DO 250 I = 1, 10
2400      J = J + 1
2410      ENCODE (FREC(I), 150) SIMCOD(J), SIMNAM(J), SIMFAC(J),
2420&                           SIMTXT(J)
2430 250  CONTINUE
2440      CONT = SECNO + 1
2450      CALL WSEC (CARD, SECNO, CONT)
2460* - - CHECK TO SEE IF ALL HAS BEEN WRITTEN.
2470      IF(J.EQ.20) RETURN
2480      GO TO 245
2490C
2500 900  PRINT, "EOF ON RWHEAD READ, TERMINATING."
```

```
2510      STOP
2520      END
2530      SUBROUTINE RSEC (DATA, SECNO, CONT, *)
2540C - - READS A 76 WORD RECORD FROM THE DATA BASE.
2550      INTEGER  SECNO, CONT
2560      CHARACTER  RECORD*456, DATA*450
2570      READ (10'SECNO, END=200, ERR=300) RECORD
2580      DECODE (RECORD, 100) DATA, CONT
2590 100  FORMAT (A450, I6)
2600      RETURN
2610 200  RETURN 1
2620 300  WRITE (06, 350) SECNO
2630 350  FORMAT (1X, "I/O ERROR WHILE READING SECTOR NUMBER",
2640&            I5, /, 1X, "PROCESSING TERMINATED.")
2650      STOP
2660      END
2670      SUBROUTINE WSEC (DATA, SECNO, CONT)
2680C - - WRITES A 76 WORD RECORD TO THE DATA BASE FILE.
2690      INTEGER  SECNO, CONT
2700      CHARACTER  RECORD*456, DATA*450
2710      ENCODE (RECORD, 100) DATA, CONT
2720 100  FORMAT (A450, I6)
2730      WRITE (10'SECNO, ERR=300) RECORD
2740      RETURN
2750 300  WRITE (06, 350) SECNO
2760 350  FORMAT (1X, "I/O ERROR WHILE WRITING TO SECTOR NUMBER",
2770&            I5, /, 1X, "PROCESSING TERMINATED.")
2780      STOP
2790      END
```

SUBROUTINE LISTR

```
1000CLISTR    LIST RECORDS.
1010      SUBROUTINE LISTR (FC)
1020C
1030C      FUNCTION CODE (FC):
1040C        0 = TREAT AS NORMAL CALL FROM 'MAIN'.
1050C        1 = TREAT AS LISTING CALL FROM 'TMOD'.
1060C
1070      CHARACTER  FNAME*6, DATES*8(3), CARD*450, REC*90(5),
1080&              SYSCOD*2(40), SYSNAM*12(40), ANS*3
1090C
1100      INTEGER  SECNO, LSIZE, DIR(3), INFO(3), DATAS(3),
1110&              CONT, DSSEC(40), DESEC(40), DWSEC(40),
1120&              NDEL(40), FIRST, FC, RNUM
1130C
1140      LOGICAL  DIREUP, INFOUP, DATAUP, WINFO
1150C
1160      EQUIVALENCE  (CARD, REC(1))
1170C
1180      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
1190C
1200      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
1210C
1220      COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DWSEC, NDEL
1230C
1240**************************************************************
1250C
1260      IF(FC.EQ.1) GO TO 90
1270      IF(.NOT. WINFO) GO TO 50
1280      PRINT, "INSTRUCTIONS? (Y OR N)"
1290      READ, ANS
1300      IF(ANS.EQ.'Y') GO TO 25
1310      WINFO = .F.; GO TO 50
1320 25   PRINT, "YOU MAY REQUEST EITHER:"
1330      PRINT, "  A - A LISTING OF ALL RECORDS IN THE FILE."
1340      PRINT, "  E - END - DONE PROCESSING."
1350      PRINT, "  T - LIST THE RECORDS ON THE TEMP FILE."
1360      PRINT, "  R - LIST A RANGE OF RECORDS (BY RECORD NUMBER)"
1370      PRINT, "      ON THE SELECT FILE."
1380      PRINT, " "
1390 50   PRINT, "ENTER A (ALL), E (END), I (INSTR), T (TEMP), OR ",
1400&           "R (RANGE)."
1410      READ, ANS
1420      IF(ANS.EQ.'A') GO TO 60
1430      IF(ANS.EQ.'R') GO TO 90
1440      IF(ANS.EQ.'I') GO TO 25
1450      IF(ANS.EQ.'T') GO TO 95
1460      IF(ANS.EQ.'E') RETURN
```

```
1470      GO TO 50
1480C
1490* - - LIST ALL OF DATA BASE CONTENTS.
1500 60   ITOTAL = 0
1510      DO 80 I = 1, DIR(3)
1520      SECNO = DSSEC(I)
1530      DO 70 J = 1, DNSEC(I)
1540      CALL RSEC (CARD, SECNO, CONT, $900)
1550      NUMB = 5
1560      CALL DSPLAY (CARD, NUMB, ITOTAL, 1, 1)
1570      SECNO = CONT
1580 70   CONTINUE
1590 80   CONTINUE
1600* - - FINISHED WITH COMPLETE DISPLAY.
1610      PRINT, "COMPLETE FILE DISPLAYED."
1620      GO TO 50
1630C
1640* - - DISPLAY RECORDS ON TEMP FILE.
1650 90   PRINT, "ENTER THE NUMBERS OF THE FIRST AND LAST RECORDS",
1660&           " TO BE DISPLAYED."
1670      READ, FIRST, LAST
1680      GO TO 100
1690 95   FIRST = 1
1700      IF(NSEL.GT.0) GO TO 97
1710 96   PRINT, "NO DATA ON TEMP FILE.."
1720      GO TO 50
1730 97   LAST = NSEL
1740 100  IF(LAST.LE.NSEL) GO TO 120
1750      PRINT 105, NSEL
1760      IF(NSEL.LE.0) GO TO 96
1770 105  FORMAT (1X, "THERE ARE ONLY", I5, " RECORDS ON THE TEMP ",
1780&            "FILE.")
1790      GO TO 90
1800C
1810 120  ITOTAL = FIRST - 1
1820      NREAD = LAST - ITOTAL
1830      IF(NREAD.GT.0) GO TO 124
1840      PRINT, "THE 'LAST' VALUE MUST BE GREATER THAN THE 'FIRST'."
1850      GO TO 90
1860 124  ITOTAL = 0
1870      IPT = 1
1880* - - COMPUTE STARTING SECTOR.
1890 125  IREM = MOD(FIRST, 5)
1900      SECNO = FIRST / 5
1910      IF(IREM.GT.0) SECNO = SECNO + 1
1920      IF(IREM.EQ.0) IREM = 5
1930* - - READ FROM TEMP FILE.
1940 127  READ (20'SECNO, END=800, ERR=850) CARD
1950* - - IF 1ST RECORD TO BE PRINTED  IS 1ST IN BUFFER, PRESS ON.
1960      IF(IREM.EQ.1) GO TO 130
1970* - - COMPUTE NUMBER TO READ.
1980      NUMB = 6 - IREM
```

```
1990      IF(NUMB.GT.NREAD) NUMB = NREAD
2000      IPT = IREM
2010      GO TO 140
2020 130  NUMB = 5
2030      IF(NREAD.LT.5) NUMB = NREAD
2040 140  CALL DSPLAY (CARD, NUMB, ITOTAL, FIRST, IPT)
2050 142  NREAD = NREAD - NUMB
2060      IF(NREAD.LE.0) GO TO 144
2070      SECNO = SECNO + 1
2080      IREM = 1
2090      GO TO 127
2100 144  PRINT 145, ITOTAL
2110 145  FORMAT (1X, "SELECTED LIST CONTAINS", I4, " RECORDS.")
2120      IF(FC.EQ.0) GO TO 50
2130      RETURN
2140C
2150 800  PRINT, "EOF WHILE READING TEMP FILE."
2160      GO TO 910
2170 850  PRINT, "ERROR RETURN WHILE READING TEMP FILE."
2180      GO TO 910
2190 900  PRINT, "EOF WHILE READING DATA BASE FILE."
2200 910  PRINT, "TERMINATING 'LIST' FUNCTION."
2210      RETURN
2220      END
```

```
0000CPUTGET  PUT/GET 'TEMP' RECORDS ON/FROM PERM-FILE.
0010C
0020C
0030      SUBROUTINE PUTGET (MODE)
0040C       MODE:
0050C         1 = PUT RECORDS ON PERM-FILE.
0060C         2 = GET RECORDS FROM PERM-FILE AND PLACE ON 'TEMP'.
0070C
0080      CHARACTER  TBUFF*450, TREC*90(5), ANS*1, TEMP*1, ATCH*30,
0090&              FNAME*6, DATES*8(3)
0100C
0110      INTEGER  SECNO, DIR(3), INFO(3), DATAS(3), RNUM
0120C
0130      LOGICAL  DIREUP, INFOUP, DATAUP, WINFO, ALL, DONE
0140C
0150      EQUIVALENCE  (TBUFF, TREC(1))
0160C
0170      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
0180C
0190      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
0200C
0210      IF(MODE.EQ.2) GO TO 30
0220      IF(NSEL.GT.0) GO TO 20
0230      PRINT, "THERE ARE NO RECORDS ON THE TEMP FILE."
0240      RETURN
0250 20   PRINT, "DO YOU WANT TO TRANSFER ALL RECORDS 'A', OR ",
0260&           "MODIFIED 'M' ONLY (A OR M)?"
0270      READ, ANS
0280      ALL = .F.
0290      IF(ANS.EQ.'A') ALL = .T.
0300 30   PRINT, "ENTER THE CAT/FILE STRING FOR YOUR SEQUENTIAL PERM",
0310&           "FILE.  YOU MUST END WITH ';'"
0320      READ, ATCH
0330      CALL DETACH (30, ISTAT,)
0340      CALL ATTACH (30, ATCH, 3, 0,,)
0350      IF(MODE.EQ.2) GO TO 100
0360C
0370* - - COMPUTE TOTAL NUMBER OF SECTORS OF DATA ON TEMP.
0380 40   IREM = MOD(NSEL, 5)
0390      NUMSEC = NSEL / 5
0400      IF(IREM.GT.0) NUMSEC = NUMSEC + 1
0410      SECNO = 1; TEMP = ' '
0420      ITOT = 0; ISTOP = 5
0430* - - READ NEXT SECTOR.
0440 45   READ (20'SECNO, END=800, ERR=850) TBUFF
0450      ITOT = ITOT + 5
0460      IF(ITOT.GT.NSEL) ISTOP = ISTOP - (ITOT-NSEL)
```

```
1470      DO 60 I = 1, ISTOP
1480      IF(ALL) GO TO 50
1490* - - EXTRACT STATUS FLAG FROM RECORD.
1500      CALL CONCAT (TEMP, 1, TREC(I), 2, 1)
1510* - - CHECK FOR MODIFIED STATUS.
1520      IF(TEMP.EQ.'M') GO TO 50
1530      GO TO 60
1540* - - WRITE RECORD TO PERM-FILE.
1550 50   WRITE (30, 55, ERR=875) TREC(I)
1560 55   FORMAT (A90)
1570 60   CONTINUE
1580* - - INCREMENT SECTOR NUMBER AND SEE IF WE SHOULD GET ANOTHER.
1590      SECNO = SECNO + 1
1600      IF(SECNO.LE.NUMSEC) GO TO 45
1610* - - PROCESSING COMPLETE.
1620 70   ENDFILE 30
1630 75   PRINT, "PROCESSING COMPLETE."
1640      RETURN
1650C
1660* - - GET MODE.
1670*     HAS TEMP FILE SPACE BEEN ALLOCATED YET?
1680 100  IF(NSEL.GE.0) GO TO 125
1690* - - NO, ALLOCATE TEMP FILE SPACE.
1700* - - DESIRED FILE CODE.
1710      LGU = 20
1720* - - SIZE, IN WORDS, FOR TEMP FILE.
1730      ISIZE = 320 * LSIZE
1740* - - ASK FOR RANDOM MASS STORAGE FILE.
1750      IMODE = 1
1760      CALL CREATE (LGU, ISIZE, IMODE, ISTAT)
1770      IF(ISTAT.EQ.0 .OR. ISTAT.EQ.5) GO TO 120
1780      PRINT, "ERROR RETURN FROM CREATE CALL."
1790      PRINT, "ERROR CODE =", ISTAT
1800      PRINT, "RETURNING TO MAIN PROGRAM."
1810      RETURN
1820C
1830* - - INITIALIZE TEMP FILE.
1840 120  CALL RANSIZ (20, 75, 1)
1850 125  NSEL = 0
1860      SECNO = 1
1870      DONE = .F.
1880 130  DO 140 I = 1, 5
1890      READ (30, 55, END=150) TREC(I)
1900      NSEL = NSEL + 1
1910 140  CONTINUE
1920      GO TO 160
1930 150  DONE = .T.
1940* - - WRITE SECTOR TO TEMP.
1950 160  WRITE (20'SECNO, ERR=900) TBUFF
1960      IF(DONE) GO TO 75
1970      SECNO = SECNO + 1
1980      GO TO 130
```

```
1990C
2000 800  PRINT, "EOF WHILE READING TEMP."
2010      GO TO 70
2020 850  PRINT, "ERROR RETURN WHILE READING TEMP."
2030      GO TO 910
2040 875  PRINT, "ERROR RETURN WHILE WRITING TO PERM-FILE."
2050      GO TO 910
2060 900  PRINT, "ERROR RETURN WHILE WRITING TO TEMP."
2070 910  PRINT, "TERMINATING PUT/GET PROCESSING."
2080      RETURN
2090      END
```

SUBROUTINE RECMOD

```
CRECMOD   ROUTINE TO MODIFY DATA RECORDS.
C
      SUBROUTINE RECMOD (RECN)
C
      CHARACTER  STR*8(15), LSTR*30, BUFF*450, SREC*90(5),
     &           NULL*6/'*NULL*'/, TEMP*6, HOLD*8, TMP*1/' '/
C
      INTEGER  RP, REC(20), FP, FLD(15), SP, RECN,
     &         ST1(18), ST2(18), CT(18)
C
      LOGICAL PNT
C
      EQUIVALENCE  (BUFF, SREC(1))
C
      COMMON /MODBUF/ BUFF
C
      COMMON /ARRAYS/ RP, REC, FP, FLD, SP, STR, LSTR, NFLDS
C
      DATA  ST1/ 1,  3,  5,  7, 15, 16, 21, 26, 31, 35, 40, 45,
     &           50, 52, 54, 84, 85, 86/,
     &       ST2/ 1,  1,  1,  1,  1,  4,  4,  4,  5,  4,  4,  4,
     &            1,  1,  1,  1,  1,  1/,
     &        CT/ 1,  2,  2,  8,  1,  5,  5,  5,  4,  5,  5,  5,
     &            2,  2, 30,  1,  1,  1/
C
      DO 30 J = 1, FP
      K = FLD(J)
      IF(K.NE.15) GO TO 5
      CALL CONCAT (TEMP, 1, LSTR, 1, 6)
      IF(TEMP.EQ.NULL) LSTR = STR(J)
      GO TO 25
    5 IF(K.LT.6 .OR. K.GT.12) GO TO 20
      IF(K.EQ.6 .OR. K.EQ.9 .OR. K.EQ.10) GO TO 8
      GO TO 18
    8 IS = 8; IC = 1; HOLD = ' '; I = 0; PNT = .F.
   10 I = I + 1
      CALL CONCAT (TMP, 1, STR(J), I, 1)
      IF(TMP.EQ.' ') GO TO 14
      IF(TMP.EQ.'.') GO TO 15
   12 CALL CONCAT (HOLD, IC, TMP, 1, 1)
      IC = IC + 1
      GO TO 16
   14 TMP = '0'
      IF(.NOT. PNT) GO TO 16
      IF(IC.LT.IS) GO TO 12
      GO TO 16
   15 PNT = .T.
```

```
1470       IS = IC + 2
1480       IF(K.EQ.9) IS = IC + 1
1490       IF(IS.GT.8) IS = 8
1500 16    IF(I.LT.IS) GO TO 10
1510       STR(J) = HOLD
1520 18    CALL RJUST (STR(J))
1530 20    CALL CONCAT (SREC(RECN), ST1(K), STR(J), ST2(K), CT(K))
1540       GO TO 30
1550 25    CALL CONCAT (SREC(RECN), ST1(K), LSTR, 1, 30)
1560 30    CONTINUE
1570       RETURN
1580       END
```

SUBROUTINE RJUST

```
0000CRJUST   SUB TO RIGHT JUSTIFY NUMERIC DATA.
0010      SUBROUTINE RJUST (TEMP)
0020C
0030* - - PLACES LEADING ZEROES IN TEMP WITH DATA RT JUSTIFIED.
0040C
0050      CHARACTER  TEMP*8, HOLD*1/' '/, OUT*8,
0060&                ZEROS*8/'00000000'/
0070C
0080      J = 0
0090* - - TRANSFER TEMP TO OUT.
0100      OUT = TEMP
0110      DO 50 I = 1, 8
0120      K = I
0130      CALL CONCAT (HOLD, 1, OUT, K, 1)
0140      IF(HOLD.EQ.' ') GO TO 70
0150      J = J + 1
0160 50   CONTINUE
0170      RETURN
0180* - - LOAD TEMP WITH ZEROES.
0190 70   TEMP = ZEROS
0200      IF(J.EQ.0) GO TO 80
0210      L = 8 - (J - 1)
0220      CALL CONCAT (TEMP, L, OUT, 1, J)
0230 80   RETURN
0240      END
```

```
C RUNSIZ  SIMSIZ SIZING ROUTINE.
C
      SUBROUTINE RUNSIZ
C
      CHARACTER  FNAME*6, DATES*8(3), CPUC*2(15), CPUN*8(15),
     &           LANGC*1(10), LANGN*8(10), SIMCOD*1(20), SIMNAM*8(20),
     &           SYSCOD*2(40), SYSNAM*12(40), ANS*1, CARD*450, AC*1,
     &           REC*90(5), FREC*42(10), SYS*2, LAN*1, CC*2, STAT*1,
     &           OAC*1, OCC*2, OLAN*1, OSYS*2, CODE*1, MMM*1/'M'/,
     &           TEMP*8(7), HOLD*8, CHA*1/' '/, YR*2, LABEL*12,
     &           LABEL1*12/'SIMPLE AVE  '/, LABEL2*12/'WEIGHTED AVE'/,
     &           CARD2*450, REC2*90(5), NC*2
C
      INTEGER  SECNO, LSIZE, DIR(3), INFO(3), DATAS(3), CONT,
     &           DSSEC(40), DESEC(40), DNSEC(40), NDEL(40), RNUM,
     &           SIMTXT(20), GINSTC(20), GINSTM(20), GDATAC(20),
     &           GDATAM(20), ST1(7), ST2(7), CT(7), TINSTC(20),
     &           TINSTM(20), TDATAC(20), TDATAM(20), DA1, DA2,
     &           TEMP2, TEMP3, TEMP5, TEMP6, TSECNO, HTHOU/100000/
C
      REAL  MAXI(20), AVETI(20), TTIMC(20), TTIMM(20), GTIMC(20),
     &           GTIMM(20), CPUS(15), LANGS(10), SIMFAC(20), TI1,
     &           CPS, TI2, CCON, LCON, TCON, WAIT(20), MAX, AVE,
     &           CPUT(15)
C
      LOGICAL  DIREUP, INFOUP, DATAUP, WINFO, FFLG(40), SFLG(20),
     &           HITF(20), WTS, OPT1, OPT2, OPT3, ADDIT, DONE, FIRST,
     &           BIG
C
      EQUIVALENCE  (CARD, REC(1), FREC(1)), (CARD2, REC2(1))
C
      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
C
      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
C
      COMMON /INSEC/ CPUC, CPUN, CPUS, CPUT, LANGC, LANGN, LANGS
C
      COMMON /INFOS/ SIMCOD, SIMNAM, SIMFAC, SIMTXT
C
      COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DNSEC, NDEL
C
      DATA ST1/ 16, 21, 26, 31, 35, 40, 45/,
     &     ST2/  4,  4,  4,  5,  4,  4,  4/,
     &     CT /  5,  5,  5,  4,  5,  5,  5/
C
C*********************************************************************
C
```

```
1470 20   IF(.NOT. WINFO) GO TO 40
1480      PRINT, "INSTRUCTIONS (Y OR N)?"
1490      READ, ANS
1500      IF(ANS.EQ.'Y') GO TO 25
1510      WINFO = .F.
1520      GO TO 40
1530 25   WINFO = .T.
1540      PRINT, "THIS ROUTINE WILL DO SIZING RUNS WITH THREE DIFFERENT ",
1550&           "OPTIONS:"
1560      PRINT, "    OPTION 1 - PERFORM A SIZING FUNCTION ON THE ENTIRE ",
1570&           "DATA BASE."
1580      PRINT, "SUBTOTALS WILL BE PRINTED AT EACH FUNCTION LEVEL FOR ",
1590&           "EACH SIMULATOR"
1600      PRINT, "CODE (SIM CODE) IN THE INFORMATION SECTION.  GRAND TOTALS ",
1610&           "BY SIM CODE, ARE"
1620      PRINT, "PRINTED AT THE END.  SIZE TOTALS IN THE INFORMATION ",
1630&           "SECTION ARE UPDATED"
1640      PRINT, "FOR EACH SIM CODE."
1650      PRINT, "    OPTION 2 - PERFORM A SIZING FUNCTION ON ALL NON-",
1660&           "DELETE MARKED RECORDS ON"
1670      PRINT, "THE SIMSIZ TEMP SELECT FILE.  AS ABOVE, SUBTOTALS ",
1680&           "AND GRAND TOTALS ARE PRINTED,"
1690      PRINT, "HOWEVER THE INFORMATION SECTION TOTALS ARE NOT AFFECTED/",
1700&           "UPDATED."
1710      PRINT, "    OPTION 3 - PERFORM A SIZING FUNCTION AGAINST THE DATA",
1720&           " BASE BY SELECTED"
1730      PRINT, "FUNCTION CODES FOR SELECTED SIM CODES.  SIZES ARE ",
1740&           "REPORTED AS AVERAGES (USING"
1750      PRINT, "DATA FROM SELECTED SIM CODES).  WEIGHTING FACTORS MAY ",
1760&           "BE APPLIED BY SIM CODE."
1770      PRINT, "COMPLEXITY FACTOR ADJUSTMENTS MAY BE MADE.  IN ADDITION, ",
1780&           "THE AVERAGE TOTALS"
1790      PRINT, "ARE AGGREGATED FOR A FINAL AVERAGE GRAND TOTAL.  NEW ",
1800&           "RECORDS, HAVING THE SIZE"
1810      PRINT, "VALUES AS DETERMINED FOR A GIVEN FUNCTION, MAY BE ",
1820&           "CREATED AND PLACED ON THE"
1830      PRINT, "SIMSIZ TEMP FILE FOR LATER ADDITION TO THE DATA BASE.  ALL ",
1840&           "THESE RECORDS WILL"
1850      PRINT, "BE MARKED WITH A USER SUPPLIED SIM CODE."
1860C
1870 40   OPT1 = .F.; OPT2 = .F.; OPT3 = .F.; OAC = ' '; OCC = '00'
1880      OLAN = ' '; OSYS = ' '; NDIR = 0; ADDIT = .F.; DONE = .F.
1890      BIG = .F.; TEMP1 = 0.0; TEMP2 = 0; TEMP3 = 0; TEMP4 = 0.0
1900      TEMP5 = 0; TEMP6 = 0; WCNT = 0.0; TSECNO = 1
1910* - - RESET TOTAL AND GRAND TOTAL TABLES, COUNTERS AND FLAGS.
1920      DO 50 I = 1, 20
1930      GINSTC(I) = 0; GINSTM(I) = 0
1940      GDATAC(I) = 0; GDATAM(I) = 0
1950      TINSTC(I) = 0; TINSTM(I) = 0
1960      TDATAC(I) = 0; TDATAM(I) = 0
1970      MAXI(I) = 0.0; AVETI(I) = 0.0
1980      TTIMC(I) = 0.0; TTIMM(I) = 0.0
```

```
1990      GTIMC(I) = 0.0; GTIMM(I) = 0.0
2000      WAIT(I) = 1.0
2010      SFLG(I) = .F.
2020      HITF(I) = .F.
2030 50   CONTINUE
2040      DO 55 I = 1, 40
2050      FFLG(I) = .F.
2060 55   CONTINUE
2070C
2080 60   PRINT, "ENTER THE NUMBER OF THE OPTION YOU CHOOSE (1 TO 3, ",
2090&           "4 WILL PRINT INSTRUCTIONS)."
2100      READ, NUMB
2110      IF(NUMB.LT.1 .OR. NUMB.GT.4) GO TO 60
2120      GO TO (100, 300, 525, 25), NUMB
2130C
2140* - - OPTION 1.
2150 100  OPT1 = .T.
2160* - - PRINT OUT HEADER.
2170      PRINT 840
2180* - - SET ALL SIM CODE AND FUNCTION FLAGS TO TRUE.
2190      DO 110 I = 1, 40
2200      FFLG(I) = .T.
2210      IF(I.GT.20) GO TO 110
2220      IF(SIMCOD(I).EQ.' ') GO TO 110
2230      SFLG(I) = .T.
2240 110  CONTINUE
2250* - - SELECT NEXT RECORD FROM THE DATA BASE.
2260 120  NDIR = NDIR + 1
2270      IF(NDIR.LE.DIR(3)) GO TO 130
2280* - - NO MORE DIRECTORY ENTRIES.
2290      DONE = .T.
2300      GO TO 350
2310* - - IF OPTION 3, CHECK TO SEE IF THIS FUNCTION HAS BEEN SELECTED.
2320 130  IF(.NOT. OPT3) GO TO 150
2330      DO 140 I = 1, 40
2340      IF(FFLG(NDIR)) GO TO 150
2350 140  CONTINUE
2360* - - THIS FUNCTION HAS NOT BEEN CHOSEN.
2370      GO TO 120
2380 150  SECNO = DSSEC(NDIR)
2390      NSEC = 1
2400* - - READ THE NEXT SECTOR OF RECORDS.
2410 155  J = 0
2420      CALL RSEC (CARD, SECNO, CONT, $900)
2430* - - 'PICK' NEXT RECORD OFF DATA BASE.
2440C
2450 160  J = J + 1
2460C
2470      IF(J.LE.5) GO TO 180
2480      SECNO = CONT
2490      NSEC = NSEC + 1
2500      IF(NSEC.LE.DNSEC(NDIR)) GO TO 155
```

```
2510* - - HAVE RUN OUT OF RECORDS FOR THIS DIRECTORY ENTRY.
2520      GO TO 120
2530* - - DECODE RECORD.
2540*     EXTRACT STATUS FLAG.
2550 180  DECODE (REC(J), 182) STAT
2560 182  FORMAT (T2, A1)
2570      IF(STAT.EQ.'D' .OR. STAT.EQ.' ') GO TO 160
2580      DECODE (REC(J), 190) AC, SYS, LAN, TI1, IN1, DA1, CPS, TI2,
2590&           IN2, DA2, CC
2600 190  FORMAT (A1, T3, A2, T15, A1, F5.2, 2I5, F4.1, F5.2, 2I5,
2610&           2I, A2)
2620* - - CHECK FOR NEW FUNCTION CODE.
2630      IF(SYS.EQ.OSYS) GO TO 197
2640      IF(OSYS.NE.'  ') GO TO 350
2650* - - RESET OSYS.
2660 192  OSYS = SYS
2670 197  IF(AC.EQ.OAC) GO TO 210
2680      DO 200 I = 1, 20
2690      IF(AC.EQ.SIMCOD(I)) GO TO 202
2700 200  CONTINUE
2710* - - THIS SIM CODE IS NOT IN THE INFO SECTION.
2720      PRINT, "THE FOLLOWING SIM CODE IS NOT IN THE INFO SECTION: ",AC
2730      PRINT, "THE RECORD WILL BE IGNORED."
2740      GO TO 160
2750 202  IF(.NOT. OPT3) GO TO 206
2760* - - IS OPT3 SO MUST CHECK FOR MATCH ON SFLG.
2770      IF(SFLG(I)) GO TO 206
2780* - - NO MATCH.
2790      OAC = '0'
2800      GO TO 160
2810* - - SAVE VALUE OF AC.
2820 206  OAC = AC
2830* - - SET APPROPRIATE HIT FLAG.
2840      HITF(I) = .T.
2850* - - SAVE VALUE OF I.
2860      IPT = I
2870* - - IF OPT2 SET SFLG.
2880      IF(OPT2) SFLG(I) = .T.
2890* - - ADJUST SIZES/TIMES BY PROPER CORRECTION FACTORS.
2900 210  IF(CC.EQ.OCC) GO TO 242
2910* - - IF NO MATCH - DEFAULT TO FACTOR OF 1.0
2920      DO 220 I = 1, 15
2930      IF(CC.EQ.CPUC(I)) GO TO 230
2940 220  CONTINUE
2950      PRINT, "COULD NOT FIND A MATCH FOR COMPUTER CODE ", CC
2960      PRINT, "USING DEFAULT VALUE OF 1.0 FOR BYTES/WORD AND TIMING ",
2970&           "CONVERSION FACTOR."
2980      CCON = 1.0
2990      CPUTM = 1.0
3000      GO TO 240
3010 230  CCON = CPUS(I)
3020      CPUTM = CPUT(I)
```

```
3030* - - SAVE CC VALUE.
3040 240  OCC = CC
3050* - - GET LANGUAGE CONVERSION FACTOR.
3060 242  IF(LAN.EQ.OLAN) GO TO 270
3070      DO 250 I = 1, 10
3080      IF(LAN.EQ.LANGC(I)) GO TO 260
3090 250  CONTINUE
3100      PRINT, "COULD NOT FIND A MATCH FOR LANGUAGE CODE ", LAN
3110      PRINT, "USING DEFAULT VALUE OF 1.0 FOR CONVERSION FACTOR."
3120      LCON = 1.0; GO TO 270
3130 260  LCON = LANGS(I)
3140**************************************************************
3150 270  TCON = LCON * SIMFAC(IPT)
3160      TTIMM(IPT) = TTIMM(IPT) + (TI1 * TCON * CPS * CPUTM)
3170      TTIMC(IPT) = TTIMC(IPT) + (TI2 * TCON * CPS * CPUTM)
3180* - - ADJUST TCON TH 'BYTES/WORD' FACTOR (CCON).
3190      TCON = TCON * CCON
3200      TINSTM(IPT) = TINSTM(IPT) + (IN1 * TCON)
3210      TINSTC(IPT) = TINSTC(IPT) + (IN2 * TCON)
3220      TDATAM(IPT) = TDATAM(IPT) + (DA1 * TCON)
3230      TDATAC(IPT) = TDATAC(IPT) + (DA2 * TCON)
3240* - - CHECK FOR GREATEST CPS VALUE.
3250      IF(CPS.GT.MAXI(IPT)) MAXI(IPT) = CPS
3260**************************************************************
3270  -   IF(OPT1 .OR. OPT3) GO TO 160
3280      GO TO 330
3290C
3300* - - HAVE CHOSEN OPTION 2.
3310 300  OPT2 = .T.
3320      NSEC = 0; ITOT = 0
3330* - - ARE THERE ANY RECORDS ON THE TEMP FILE?
3340      IF(NSEL.GT.0) GO TO 310
3350      PRINT, "NO RECORDS ON THE SELECT TEMP FILE."
3360      OPT2 = .F.
3370      GO TO 60
3380* - - SET FOR THE FIRST SECTOR.
3390 310  SECNO = 1
3400* - - PRINT OUT HEADER.
3410      PRINT 840
3420      IR = MOD(NSEL, 5)
3430      ISEC = NSEL / 5
3440      IF(IR.GT.0) ISEC = ISEC + 1
3450      IF(IR.EQ.0) IR = 5
3460* - - READ FROM THE TEMP FILE.
3470 320  READ (20'SECNO, END=910, ERR=920) CARD
3480      J = 0
3490      NSEC = NSEC + 1
3500      ISTOP = IR
3510      IF(NSEC.LT.ISEC) ISTOP = 5
3520C
3530 330  J = J + 1
3540C
```

```
3550      IF(J.GT.ISTOP) GO TO 335
3560* - - CHECK TO SEE IF WE ARE AT THE END OF THE TEMP FILE.
3570      ITOT = ITOT + 1
3580      IF(ITOT.LE.NSEL) GO TO 180
3590      GO TO 340
3600* - - READ NEXT SECTOR.
3610 335  IF(NSEC.EQ.ISEC) GO TO 340
3620      SECNO = SECNO + 1
3630      GO TO 320
3640* - - HAVE REACHED END OF DATA.
3650 340  DONE = .T.
3660      GO TO 350
3670C
3680* - - HAVE DETECTED SHIFT IN FUNCTION CODE OR END OF DATA.
3690 350  IF(.NOT. OPT3) GO TO 480
3700      LABEL = LABEL1
3710      IF(WTS) LABEL = LABEL2
3720* - - ADJUST TOTALS BY WEIGHTS (AS NECESSARY).
3730      DO 360 I = 1, 20
3740      IF(.NOT. HITF(I)) GO TO 360
3750      TEMP1 = TEMP1 + (TTIMM(I) * WAIT(I))
3760      TEMP2 = TEMP2 + (TINSTM(I) * WAIT(I))
3770      TEMP3 = TEMP3 + (TDATAM(I) * WAIT(I))
3780      TEMP4 = TEMP4 + (TTIMC(I) * WAIT(I))
3790      TEMP5 = TEMP5 + (TINSTC(I) * WAIT(I))
3800      TEMP6 = TEMP6 + (TDATAC(I) * WAIT(I))
3810 360  CONTINUE
3820* - - FIND MAX ITERATION RATE.
3830      MAX = 0.0
3840      DO 370 I = 1, 20
3850      IF(MAXI(I).LE.MAX) GO TO 370
3860      MAX = MAXI(I)
3870 370  CONTINUE
3880      IF(TEMP4.GT.0.0) GO TO 372
3890      AVE = 0.0; GO TO 374
3900 372  AVE = TEMP4 / MAX
3910 374  PRINT, " "
3920      PRINT, " FUNCTION ", OSYS
3930      PRINT 375, LABEL, MAX, AVE, TEMP4, TEMP5, TEMP6,
3940&           TEMP1, TEMP2, TEMP3
3950 375  FORMAT (2X, A12, F8.2, F9.2, F9.2, I8, I7, F9.2, I8, I7)
3960      PRINT, "ENTER COMPLEXITY ADJUSTMENT."
3970      READ, CMPLX
3980      IF(CMPLX.LT.0.0005) CMPLI = 1.0
3990* - - MULTIPLY VALUE BY COMPLEXITY FACTOR.
4000      TEMP1 = TEMP1 * CMPLI
4010      TEMP2 = TEMP2 * CMPLX
4020      TEMP3 = TEMP3 * CMPLX
4030      TEMP4 = TEMP4 * CMPLX
4040      TEMP5 = TEMP5 * CMPLI
4050      TEMP6 = TEMP6 * CMPLX
4060      TEMP7 = AVE * CMPLX
```

```
      IF(.NOT. ADDIT) GO TO 440
      PRINT, "ADD TO TEMP (Y OR N)?"
      READ, ANS
      IF(ANS.EQ.'N') GO TO 440
- - CREATE A RECORD AND ADD TO THE TEMP FILE.
      REC2(JPT) = ' '
      IF(TEMP1.LT.0.0005) GO TO 370
- - ADJUST MFG'S TIMING TO GET AVERAGE TIMING/CYCLE.
      TEMP1 = TEMP1 / MAX
- - CHECK FOR TOO LARGE FIELDS.
      IF(TEMP2.GE.HTHOU .OR. TEMP3.GE.HTHOU .OR.
     &   TEMP5.GE.HTHOU .OR. TEMP6.GE.HTHOU) BIG = .T.
      IF(.NOT. BIG) GO TO 370
- - DIVIDE BY 5.
      IF(TEMP2.GT.0) TEMP2 = TEMP2 / 5
      IF(TEMP3.GT.0) TEMP3 = TEMP3 / 5
      IF(TEMP5.GT.0) TEMP5 = TEMP5 / 5
      IF(TEMP6.GT.0) TEMP6 = TEMP6 / 5
370   ENCODE (TEMP(1), 380) TEMP1
380   FORMAT (F6.2, 2X)
      ENCODE (TEMP(2), 390) TEMP2
390   FORMAT (I5, 3X)
      ENCODE (TEMP(3), 390) TEMP3
      ENCODE (TEMP(5), 380) TEMP7
      ENCODE (TEMP(6), 390) TEMP5
      ENCODE (TEMP(7), 390) TEMP6
      ENCODE (TEMP(4), 400) MAX
400   FORMAT (F5.1, 3X)
      DO 420 K = 1, 7
      HOLD = ' '
      ICNT = 0
      DO 410 L1 = 1, 8
      L = L1
      CALL CONCAT (CHA, 1, TEMP(K), L, 1)
      IF(CHA.EQ.' ' .OR. CHA.EQ.'.') GO TO 410
      ICNT = ICNT + 1
      CALL CONCAT (HOLD, ICNT, CHA, 1, 1)
410   CONTINUE
      CALL RJUST (HOLD)
      CALL CONCAT (REC2(JPT), ST1(K), HOLD, ST2(K), CT(K))
420   CONTINUE
      DECODE (DATES(3), 425) YR
425   FORMAT (T7, A2)
      MC = '  '
      IF(BIG) MC = 'M1'
      ENCODE (REC2(JPT), 430) CODE, MMM, OSYS, YR, MC
430   FORMAT (2A1, A2, 10X, 'F', T50, 2A2, T84, '000')
      NSEL = NSEL + 1
      JPT = JPT + 1
- - IF LESS THAN FULL SECTOR, PROCESS NEXT RECORD.
      IF(JPT.LE.5) GO TO 440
- - WRITE SECTOR TO TEMP.
```

```
4590 436  WRITE (20'TSECNO, ERR=930) CARD2
4600      IF(FIRST) GO TO 725
4610      TSECNO = TSECNO + 1
4620      JPT = 1
4630* - - ADD TOTALS TO GRAND TOTALS.
4640*     BUT FIRST CHECK FOR BIG AND MAKE COUNTERCORRECTION AS NEEDED.
4650 440  ICOR = 1
4660      IF(BIG) ICOR = 5
4670      BIG = .F.
4680      GTIMM(1) = GTIMM(1) + TEMP1
4690      GINSTM(1) = GINSTM(1) + TEMP2 * ICOR
4700      GDATAM(1) = GDATAM(1) + TEMP3 * ICOR
4710      GTIMC(1) = GTIMC(1) + TEMP4
4720      GINSTC(1) = GINSTC(1) + TEMP5 * ICOR
4730      GDATAC(1) = GDATAC(1) + TEMP6 * ICOR
4740 442  IF(.NOT. DONE) GO TO 446
4750      PRINT, "REQUESTED SIZING COMPLETE.  WOULD YOU LIKE TO DO SOME",
4760&           " MORE (Y OR N)?"
4770      READ, ANS
4780      IF(ANS.EQ.'Y') GO TO 446
4790      FIRST = .T.
4800 444  IF(DONE .AND. ADDIT .AND. (NSEL.GT.0) .AND. (JPT.GT.1)) GO TO 436
4810      IF(DONE) GO TO 725
4820* - - RESET TABLES.
4830 446  DO 450 K = 1, 20
4840      IF(.NOT. HITF(K)) GO TO 450
4850      TINSTC(K) = 0; TINSTM(K) = 0
4860      TDATAC(K) = 0; TDATAM(K) = 0
4870      MAXI(K) = 0.0
4880      AVETI(K) = 0.0
4890      TTIMC(K) = 0.0; TTIMM(K) = 0.0
4900      HITF(K) = .F.
4910 450  CONTINUE
4920      TEMP1 = 0.0; TEMP2 = 0
4930      TEMP3 =   0; TEMP4 = 0.0
4940      TEMP5 =   0; TEMP6 = 0
4950      PRINT, " "
4960* - - INSURE AC NE OAC.
4970      OAC = '0'
4980      IF(.NOT. DONE) GO TO 192
4990* - - RESET FLAGS.
5000      DO 460 I = 1, 40
5010      FFLG(I) = .F.
5020      IF(I.GT.20) GO TO 460
5030      SFLG(I) = .F.
5040      WAIT(I) = 1.0
5050      HITF(I) = .F.
5060 460  CONTINUE
5070      OCC = '00'; OLAN = ' '; OSYS = ' '; DONE = .F.; NDIR = 0
5080      WCNT = 0.0
5090      GO TO 540
5100C
```

```
5110* - - PROCESS TOTALS FOR OPT 1 AND 2.
5120 480  PRINT, " FUNCTION ", OSYS
5130      DO 500 I = 1, 20
5140      IF(.NOT. HITF(I)) GO TO 500
5150      ENCODE (LABEL, 490) SIMNAM(I)
5160 490  FORMAT ("FOR ", A8)
5170      IF(TTIMC(I).GT.0.0) GO TO 492
5180      AVE = 0.0; GO TO 494
5190 492  AVE = TTIMC(I) / MAXI(I)
5200 494  PRINT 375, LABEL, MAXI(I), AVE, TTIMC(I), TINSTC(I),
5210&           TDATAC(I), TTIMM(I), TINSTM(I), TDATAM(I)
5220* - - ADD TOTALS TO GRAND TOTALS.
5230      GTIMC(I) = GTIMC(I) + TTIMC(I)
5240      GTIMM(I) = GTIMM(I) + TTIMM(I)
5250      GINSTC(I) = GINSTC(I) + TINSTC(I)
5260      GINSTM(I) = GINSTM(I) + TINSTM(I)
5270      GDATAC(I) = GDATAC(I) + TDATAC(I)
5280      GDATAM(I) = GDATAM(I) + TDATAM(I)
5290* - - RESET TOTALS AND FLAGS.
5300      TTIMC(I) = 0.0; TTIMM(I) = 0.0
5310      TINSTC(I) = 0; TINSTM(I) = 0
5320      TDATAC(I) = 0; TDATAM(I) = 0
5330      HITF(I) = .F.; MAXI(I) = 0.0
5340 500  CONTINUE
5350* - - INSURE AC NE OAC.
5360      OAC = '0'
5370      PRINT, " "
5380* - - IF DONE, GO TO GRAND TOTALS.
5390      IF(DONE) GO TO 725
5400* - - PRESS ON WITH PROCESSING LATEST RECORDS.
5410      GO TO 192
5420C
5430* - - SELECT OPTION 3.
5440 525  OPT3 = .T.
5450      FIRST = .T.
5460      PRINT, "DO YOU WANT TO ANALYZE ALL THE CURRENT FUNCTION ",
5470&           "CODES (Y OR N)?"
5480      READ, ANS
5490      IF(ANS.EQ.'N') GO TO 540
5500      IF(ANS.NE.'Y') GO TO 525
5510* - - TURN ON ALL FUNCTION FLAGS.
5520      DO 530 I = 1, DIR(3)
5530      FFLG(I) = .T.
5540 530  CONTINUE
5550      GO TO 570
5560* - - TURN ON ONLY SELECTED FLAGS.
5570 540  PRINT, "ENTER THE 2 CHARACTER FUNCTION CODES (ONE SET PER LINE) ",
5580&           "THAT YOU WISH TO"
5590      PRINT, "ANALYZE.  END WITH '00'."
5600 545  READ, STS
5610      IF(STS.EQ.'00') GO TO 570
5620      DO 550 I = 1, DIR(3)
```

```
5630      IF(SYS.EQ.SYSCOD(I)) GO TO 560
5640 550  CONTINUE
5650      PRINT, "COULD NOT FIND MATCH FOR FUNCTION CODE ", SYS
5660      PRINT, "ENTER NEXT VALUE."
5670      GO TO 545
5680C - - TURN ON FLAG.
5690 560  FFLG(I) = .T.
5700      GO TO 545
5710C
5720 570  PRINT, "THESE ARE THE FUNCTIONS WHICH WILL BE ANALYZED:"
5730      DO 580 I = 1, DIR(3)
5740      IF(.NOT. FFLG(I)) GO TO 580
5750      PRINT 575, SYSCOD(I), SYSNAM(I)
5760 575  FORMAT (1X, A2, 2X, A12)
5770 580  CONTINUE
5780      PRINT, "DO YOU WANT TO ALTER THE LIST (Y OR N)?"
5790      READ, ANS
5800      IF(ANS.EQ.'N') GO TO 590
5810      DO 585 I = 1, DIR(3)
5820      FFLG(I) = .F.
5830 585  CONTINUE
5840      PRINT, "START OVER AND"
5850      GO TO 540
5860 590  IF(.NOT. WINFO) GO TO 592
5870      PRINT, "YOU MUST NOW SPECIFY THE SIMULATOR CODES OF THE ",
5880&           "SYSTEMS YOU WISH TO"
5890      PRINT, "INCLUDE IN THE ANALYSIS.  THE SIZING VALUES ",
5900&           "WILL BE AVERAGED, BY CATAGORY"
5910      PRINT, "(TIMING, INSTRUCTION SIZE, DATA SIZE, ETC), FOR EACH ",
5920&           "SIM CODE AND THE AVERAGE"
5930      PRINT, "SIZES WILL BE PRINTED OUT BY MAJOR FUNCTION.  THIS ",
5940&           "AVERAGING MAY BE 'WEIGHTED' "
5950      PRINT, "SO THAT THE VALUES FOR THE VARIOUS SIMULATOR SYS",
5960&           "TEMS (SIM CODES) WILL HAVE "
5970      PRINT, "MORE OR LESS IMPACT ON THE FINAL AVERAGES.  (SEE ",
5980&           "USER'S MANUAL FOR FURTHER "
5990      PRINT, "EIPLAINATION OF WEIGHTING.)"
6000 592  PRINT, "DO YOU WISH TO WEIGHT THE SIZING DATA (Y OR N)?"
6010      READ, ANS
6020      IF(ANS.EQ.'N') GO TO 620
6030      WTS = .T.
6040 594  PRINT, "ENTER 1 CHARACTER SIM CODE AND ITS WEIGHT (SEPERATED ",
6050&           "BY A COMMA). "
6060      PRINT, "ENTER '$, 0' TO SIGNIFY END."
6070 596  READ, AC, TEMP1
6080      IF(AC.EQ.'$') GO TO 660
6090C - - LOOK FOR MATCH ON THE SIM CODE.
6100      DO 600 I = 1, 20
6110      IF(AC.EQ.SIMCOD(I)) GO TO 610
6120 600  CONTINUE
6130      PRINT, "COULD NOT FIND A MATCH FOR SIM CODE ", AC
6140      PRINT, "ENTER NEXT SET OF VALUES."
```

```
6150      GO TO 596
6160* - - SET FLAG AND WEIGHT.
6170 610  SFLG(I) = .T.
6180      WAIT(I) = TEMP1
6190      WCNT = WCNT + WAIT(I)
6200      GO TO 596
6210C
6220* - - ENTER SIM CODES ONLY.
6230 620  PRINT, "ENTER THE 1 CHARACTER SIM CODES (ONE PER LINE)."
6240      PRINT, "END WITH '#'."
6250      WTS = .F.
6260 630  READ, AC
6270      IF(AC.EQ.'#') GO TO 660
6280* - - LOOK FOR MATCH ON SIM CODE.
6290      DO 640 I = 1, 20
6300      IF(AC.EQ.SIMCOD(I)) GO TO 650
6310 640  CONTINUE
6320      PRINT, "COULD NOT FIND A MATCH FOR SIM CODE ", AC
6330      PRINT, "ENTER NEXT VALUE."
6340      GO TO 630
6350* - - SET SIM CODE FLAG.
6360 650  SFLG(I) = .T.
6370      WAIT(I) = 1.0
6380      WCNT = WCNT + 1.0
6390      GO TO 630
6400* - - LIST SELECTIONS.
6410 660  PRINT, "THESE ARE THE SIMULATOR SYSTEMS YOU HAVE CHOSEN TO ",
6420&           "WORK WITH:"
6430      DO 670 I = 1, 20
6440      IF(.NOT. SFLG(I)) GO TO 670
6450      IF(WTS) GO TO 666
6460      PRINT 662, SIMCOD(I), SIMNAM(I)
6470 662  FORMAT (1X, A1, 2X, A8)
6480      WAIT(I) = WAIT(I) / WCNT
6490      GO TO 670
6500 666  IF(WAIT(I).GT.0.0) GO TO 667
6510      GO TO 6672
6520 667  WAIT(I) = WAIT(I) / WCNT
6530 6672 PRINT 668, SIMCOD(I), SIMNAM(I), WAIT(I)
6540 668  FORMAT (1X, A1, 2X, A8, "  WEIGHT =", F5.2)
6550 670  CONTINUE
6560      PRINT, "DO YOU WISH TO CHANGE THESE VALUES (Y OR N)?"
6570      READ, ANS
6580      IF(ANS.EQ.'N') GO TO 690
6590      PRINT, "START OVER, ENTER ALL VALUES AGAIN."
6600      DO 680 I = 1, 20
6610      SFLG(I) = .F.
6620 680  CONTINUE
6630      WCNT = 0.0
6640      IF(WTS) GO TO 594
6650      GO TO 620
6660 690  IF(.NOT. FIRST) GO TO 692
```

```
6670      PRINT, "WILL YOU WANT TO BE ADDING RECORDS TO THE 'TEMP' ",
6680&           "FILE (Y OR N)?"
6690      READ, ANS
6700      FIRST = .F.
6710      IF(ANS.EQ.'Y') GO TO 694
6720 692  PRINT 840
6730      GO TO 120
6740* - - CHECK TO SEE IF THE TEMP FILE IS ATTACHED.
6750 694  IF(NSEL.GE.0) GO TO 700
6760* - - MUST ATTACH TEMP FILE.  DESIRED FILE CODE:
6770      LGU = 20
6780* - - SIZE, IN WORDS, FOR TEMP FILE.
6790      ISIZE = 320 * LSIZE
6800* - - ASK FOR RANDOM MASS STORAGE FILE.
6810      MODE = 1
6820      CALL CREATE (LGU, ISIZE, MODE, ISTAT)
6830      IF(ISTAT.EQ.0 .OR. ISTAT.EQ.5) GO TO 696
6840      PRINT, "ERROR RETURN FROM CREATE CALL.  ERROR CODE =", ISTAT
6850      PRINT, "RETURNING TO MAIN PROGRAM."
6860      RETURN
6870* - - INITIALIZE TEMP FILE.
6880 696  CALL RANSIZ (20, 75, 1)
6890 700  NSEL = 0
6900      SECNO = 1
6910      JPT = 1; ADDIT = .T.
6920 705  PRINT, "ENTER 1 CHARACTER SIM CODE TO BE APPLIED TO ALL ",
6930&           "RECORDS."
6940      READ, CODE
6950      DO 710 I = 1, 20
6960      IF(CODE.EQ.SIMCOD(I)) GO TO 715
6970 710  CONTINUE
6980      GO TO 692
6990 715  PRINT 720, CODE, SIMNAM(I)
7000 720  FORMAT (1X, "SIM CODE ", A1, " IS ALREADY IN USE FOR THE ",
7010&           A12, " SIMULATOR.",/, "DO YOU STILL WISH TO USE ",
7020&           "THAT SIM CODE (Y OR N)?")
7030      READ, ANS
7040      IF(ANS.EQ.'N') GO TO 705
7050      PRINT 840
7060      GO TO 120
7070C
7080* - - DONE READING RECORDS.  PRINT GRAND TOTALS.
7090 725  PRINT 730
7100 730  FORMAT (1H0, 27(1H*), " G R A N D  T O T A L S ", 27(1H*), //)
7110      PRINT 840
7120      PRINT, " "
7130      IF(.NOT. OPT3) GO TO 760
7140      LABEL = LABEL1
7150      IF(WTS) LABEL = LABEL2
7160      PRINT 740, LABEL, CTIMC(1), CINSTC(1), CDATAC(1), CTIMM(1),
7170&           CINSTM(1), CDATAM(1)
7180 740  FORMAT (1X, A12, 4X, "---", 7X, "---", 3X, F7.2, I8, I7,
```

```
7190&           F9.2, I8, I7)
7200      PRINT, " "
7210      PRINT, " "
7220      IF(.NOT. ADDIT) RETURN
7230      PRINT 750, NSEL
7240 750  FORMAT (1X, I3, " RECORDS WERE ADDED TO THE TEMP FILE.", ///)
7250      RETURN
7260 760  DO 770 I = 1, 20
7270      IF(.NOT. SFLG(I)) GO TO 770
7280      ENCODE (LABEL, 490) SIMNAM(I)
7290      PRINT 740, LABEL, CTIMC(I), CINSTC(I), CDATAC(I), CTIMM(I),
7300&           CINSTM(I), CDATAM(I)
7310 770  CONTINUE
7320      IF(OPT2) GO TO 795
7330* - - UPDATE VALUES IN INFO SECTION.
7340      SECNO = INFO(1) + 1
7350      J = 0
7360 772  CALL RSEC (CARD, SECNO, CONT, $900)
7370      DO 790 I = 1, 10
7380      J = J + 1
7390      ENCODE (FREC(I), 780) CINSTC(J), CDATAC(J), CTIMC(J)
7400 780  FORMAT (T19, 2I7, F8.2)
7410 790  CONTINUE
7420      CALL WSEC (CARD, SECNO, CONT)
7430      IF(J.EQ.20) GO TO 795
7440      SECNO = CONT
7450      GO TO 772
7460 795  PRINT, " "
7470      PRINT, " "
7480      RETURN
7490C
7500* - - OUTPUT HEADER.
7510 840  FORMAT (1H0, 15X, "MAXIMUM    AVE   ----CURRENT TOTAL",
7520&           "-----  --MANUFACTURER'S EST--", /, 15X,
7530&           "ITERATION  TIME/    TIME    INSTR   DATA    TIME",
7540&           "    INSTR   DATA", /, 17X, "RATE     CYCLE  MSEC/",
7550&           "SEC  BYTES  BYTES  MSEC/SEC  BYTES  BYTES")
7560C
7570 900  PRINT, "ERROR RETURN WHILE READING FROM DATA BASE."
7580      GO TO 940
7590 910  PRINT, "EOF WHILE READING TEMP FILE."
7600      GO TO 940
7610 920  PRINT, "I/O ERROR WHILE READING TEMP FILE."
7620      GO TO 940
7630 930  PRINT, "I/O ERROR WHILE WRITING TO TEMP FILE."
7640 940  PRINT, "RETURNING TO MAIN PROGRAM."
7650      RETURN
7660      END
```

SUBROUTINE SELREC

```
1000CSELREC   SELECT RECORDS.
1010      SUBROUTINE SELREC
1020C
1030      CHARACTER  FNAME*6, DATES*8(3), CARD*450, REC*90(5),
1040&                SYSCOD*2(40), SYSNAM*12(40), FL1*1,
1050&                SEL*1/'S'/, GEE*1/'G'/, ANS*3, BUFF*450,
1060&                SREC*90(5)
1070C
1080      INTEGER  SECNO, LSIZE, DIR(3), INFO(3), DATAS(3),
1090&              CONT, DSSEC(40), DESEC(40), DNSEC(40),
1100&              NDEL(40), OLSEC, RNUM, TSEC
1110C
1120      LOGICAL  HIT/.F./, REFIND/.F./, WINFO
1130C
1140      EQUIVALENCE  (CARD, REC(1)), (BUFF, SREC(1))
1150C
1160      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
1170C
1180      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
1190C
1200      COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DNSEC, NDEL
1210C
1220      IF(.NOT. WINFO) GO TO 15
1230      PRINT, "DO YOU WANT SELECTION INSTRUCTIONS (Y OR N)?"
1240      READ, ANS
1250      IF(ANS.EQ.'Y') GO TO 10
1260      WINFO = .F.; GO TO 15
1270 10   PRINT, "THIS ROUTINE SELECTS RECORDS FROM THE DATA BASE"
1280      PRINT, "AND WRITES THEM TO THE SIMSIZ TEMP FILE."
1290      PRINT, "THE FILE MAY THEN BE LISTED OR RECORDS MAY BE "
1300      PRINT, "MODIFIED, DELETED, OR RUN THRU THE SIZING"
1310      PRINT, "FUNCTION.  THIS ROUTINE GIVES YOU THE OPTION"
1320      PRINT, "OF CHOOSING:"
1330      PRINT, "  N - NEW SELECTION - SELECT RECORDS BASED ON"
1340      PRINT, "      NEW SELECTION CRITERIA, PLACE THEM ON THE"
1350      PRINT, "      TEMP FILE, AND MARK THEM AS SELECTED IN"
1360      PRINT, "      THE DATA BASE.  ANY RECORDS IN THE DATA"
1370      PRINT, "      BASE WHICH DO NOT MEET THIS NEW CRITERIA,"
1380      PRINT, "      EVEN IF THEY WERE PREVIOUSLY MARKED AS"
1390      PRINT, "      SELECTED, ARE DE-SELECTED IN THE DATA"
1400      PRINT, "      BASE."
1410      PRINT, " "
1420      PRINT, "  P - PREVIOUS SELECTION - LOAD THE TEMP FILE"
1430      PRINT, "      WITH RECORDS WHICH HAD PREVIOUSLY BEEN"
1440      PRINT, "      SELECTED FROM THE DATA BASE."
1450      PRINT, " "
1460      PRINT, "  E - END PROCESSING - RETURN."
```

```
1470      PRINT, " "
1480      WINFO = .T.
1490 15   PRINT, "ENTER E (END), I (INSTRUCTIONS), N (NEW), ",
1500&           "P (PREVIOUS)."
1510      READ, ANS
1520      REFIND = .F.
1530      IF(ANS.EQ.'I') GO TO 10
1540      IF(ANS.EQ.'N') GO TO 17
1550      IF(ANS.EQ.'P') REFIND = .T.
1560      IF(ANS.EQ.'E') RETURN
1570      IF(.NOT. REFIND) GO TO 15
1580      GO TO 20
1590C
1600* - - OBTAIN NEW SELECTION CRETERIA.
1610 17   CALL CRITER
1620* - - CREATE AND ATTACH TEMP FILE.
1630 20   IF(NSEL.GE.0) GO TO 23
1640* - - DESIRED FILE CODE.
1650      LGU = 20
1660* - - SIZE, IN WORDS, FOR TEMP FILE.
1670      ISIZE = 320 * LSIZE
1680* - - ASK FOR RANDOM MASS STORAGE FILE.
1690      MODE = 1
1700      CALL CREATE (LGU, ISIZE, MODE, ISTAT)
1710      IF(ISTAT.EQ.0 .OR. ISTAT.EQ.5)GO TO 23
1720      PRINT, "ERROR RETURN FROM CREATE CALL."
1730      PRINT, "ERROR CODE =", ISTAT
1740      PRINT, "RETURNING TO MAIN PROGRAM."
1750      RETURN
1760* - - REWIND TEMP FILE.
1770 23   NSEL = 0
1780      IC = 0; TSEC = 1
1790      CALL RANSIZ (20, 75, 1)
1800C
1810      DO 50 I = 1, DIR(3)
1820      SECNO = DSSEC(I)
1830C
1840      DO 40 J = 1, DNSEC(I)
1850      CALL RSEC (CARD, SECNO, CONT, $900)
1860      HIT = .F.
1870C
1880      DO 35 K = 1, 5
1890      DECODE (REC(K), 25) FL1
1900 25   FORMAT (T2, A1)
1910      IF(FL1.EQ.' ' .OR. FL1.EQ.'D') GO TO 35
1920      IF(.NOT. REFIND) GO TO 26
1930      IF(FL1.EQ.'S') GO TO 28
1940* - - PRESS ON WITH SEARCH.
1950      GO TO 35
1960 26   CALL EVAL (REC(K), ISTAT)
1970      IF(ISTAT.EQ.1) GO TO 33
1980* - - MARK RECORD FOR SELECTION.
```

```
1990      ENCODE (REC(K), 25) SEL
2000 28   HIT = .T.
2010* - - TRANSFER RECORD TO OUTPUT BUFFER.
2020      IC = IC + 1
2030      SREC(IC) = REC(K)
2040      IF(IC.LT.5) GO TO 30
2050* - - WRITE BUFF TO TEMP FILE.
2060      WRITE (20'TSEC, ERR=800) BUFF
2070      IC = 0
2080      TSEC = TSEC + 1
2090      IF(TSEC.LT.425) GO TO 30
2100      PRINT, "TEMP FILE IS FULL.  TERMINATING SELECTION."
2110      NSEL = NSEL + 1; GO TO 70
2120* - - INCREMENT SELECTED-RECORD COUNTER.
2130 30   NSEL = NSEL + 1
2140      GO TO 35
2150* - - MARK AS NOT SELECTED.
2160 33   ENCODE (REC(K), 25) CEE
2170      HIT = .T.
2180 35   CONTINUE
2190      IF(CONT.EQ.0) GO TO 37
2200* - - SAVE OLD SECTOR NUMBER.
2210      OLSEC = SECNO
2220      SECNO = CONT
2230 37   IF(.NOT. HIT) GO TO 40
2240* - - WRITE MARKED RECORDS BACK TO FILE.
2250*     RESET SECNO BEFORE WRITE.
2260      IF(CONT.NE.0) SECNO = OLSEC
2270      CALL WSEC (CARD, SECNO, CONT)
2280      SECNO = CONT
2290 40   CONTINUE
2300 50   CONTINUE
2310      PRINT 60, NSEL
2320 60   FORMAT (1X, "SEARCH OF DATA BASE COMPLETE.",/, 1X,
2330&            I4, " RECORDS SELECTED.")
2340      IF(NSEL.EQ.0) GO TO 70
2350      IF(IC.EQ.0) GO TO 70
2360      WRITE (20'TSEC, ERR=800) BUFF
2370 70   RETURN
2380C
2390 800  PRINT, "ERROR RETURN WHILE WRITING TO TEMP FILE."
2400      GO TO 910
2410C
2420 900  PRINT, "EOF WHILE READING DB IN SELECT."
2430 910  PRINT, "TERMINATING FUNCTION."
2440      RETURN
2450      END
```

```
1000CSIMBLD   INITIALIZE HEADER FOR DATA BASE FILE.
1010C
1020      SUBROUTINE SIMBLD
1030C
1040      CHARACTER  FNAME*6, DATES*8(3), CARD*450/' '/, ATCH*20,
1050&                TODAT*8, SYSCOD*2(40), SYSNAM*12(40),
1060&                NAM*12(27), COD*2(27), CPUC*2(15), CPUN*8(15),
1070&                LANGC*1(10), LANGN*8(10), SIMCOD*1(20),
1080&                SIMNAM*8(20)
1090C
1100      INTEGER  SECNO, LSIZE, DIR(3), INFO(3), DATAS(3),
1110&              CONT, DSSEC(40), DESEC(40), DNSEC(40),
1120&              NDEL(40), RNUM, NDIR/27/, SIMTXT(20)
1130C
1140      REAL  CPUS(15), CPUT(15), LANGS(10), SIMFAC(20)
1150C
1160      LOGICAL  DIREUP, INFOUP, DATAUP, WINFO
1170C
1180      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
1190C
1200      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
1210C
1220      COMMON /INSEC/ CPUC, CPUN, CPUS, CPUT, LANGC, LANGN, LANGS
1230C
1240      COMMON /INFOS/ SIMCOD, SIMNAM, SIMFAC, SIMTXT
1250C
1260      COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DNSEC, NDEL
1270C
1280      DATA COD/'AC', 'AP', 'AS', 'AV', 'CD', 'CM',
1290&              'CN', 'CR', 'DF', 'EG', 'EP', 'EV',
1300&              'EW', 'FC', 'FS', 'GP', 'HD', 'IN',
1310&              'LG', 'MO', 'NV', 'RD', 'SE', 'SS',
1320&              'TC', 'VR', 'VS'/
1330      DATA NAM/'AURAL CUES  ', 'AUX/EXT POWR', 'ARMA/STORES ',
1340&              'AVIONICS    ', 'COCPIT DISPL', 'COMMUNICATON',
1350&              'COMP,NON-REA', 'COMP,REAL TI', 'DATA FILES  ',
1360&              'ENGINE(S)   ', 'ELEC POWER  ', 'ENVIRONMENTL',
1370&              'ELEC WARFARE', 'FLT CONT SUR', 'FUEL SYSTEMS',
1380&              'GROWTH PROVI', 'HYDRAULICS  ', 'INSTRUCTIONL',
1390&              'LANDING GEAR', 'MOTION      ', 'NAVIGATION  ',
1400&              'RADAR       ', 'SIM ENVIRONT', 'SPECL SYSTMS',
1410&              'TACTICS     ', 'VIS,REAL TIM', 'VIS, SUPPORT'/
1420C
1430      CHARACTER  CPUCB*2(15), CPUNB*8(15), LANGCB*1(10),
1440&                LANGNB*8(10), SIMCODB*1(20), SIMNAMB*8(20)
1450C
1460      REAL   CPUSB(15), CPUTB(15), LANGSB(10), SIMFACB(20)
```

```
1470C
1480      INTEGER  SIMTXTB(20)
1490C
1500      DATA CPUCB /'W1', 'W2', 'B1', 'W3', 'W4', 'B2', 'W5',
1510&            'M1', 'W6', 6 * 1H /,
1520&          CPUNB /'SEL 840 ', 'SEL 840A', 'INTERDAT', 'SEL32/35',
1530&            'HARRIS/5', 'UNITY   ', 'SEL32/55', 'MULTI BY',
1540&            'INTR32/?', 6 * 1H /,
1550&          CPUSB / 3.0, 3.0, 1.0, 4.0, 3.0, 1.0, 4.0, 5.0, 4.0,
1560&            6 * 1.0/,
1570&          CPUTB / 15 * 1.0/
1580C
1590      DATA LANGCB /'F', 'A', 'C', 'P', 'M', 'U', 'D', 3 * 1H  /,
1600&          LANGNB /'FORTRAN', 'ASEMBLER', 'CMAP', 'PLACE',
1610&            'MIXED', 'UNKNOWN', 'DATAONLY', 3 * 1H  /,
1620&          LANGSB /10*1.0/
1630C
1640      DATA SIMCODB /'A', 'B', 'C', 'D', 'E', 'J', 'M', 13 * 1H  /,
1650&          SIMNAMB /'C-141', 'C-5', 'C-130', 'C-141CPT', 'C-5CPT',
1660&            'F-16', 'A-10', 13 * 1H  /,
1670&          SIMFACB /20 * 1.0/,
1680&          SIMTXTB /20 * 0/
1690C
1700*****************************************************************
1710C
1720      PRINT, "ENTER THE CAT/FILE STRING FOR THE FILE TO BE ",
1730&           "BUILT.  END WITH ';'."
1740      READ, ATCH
1750      CALL DETACH (10, ISTAT,)
1760      CALL ATTACH (10, ATCH, 3, 1,,)
1770      CALL RANSIZ (10, 76, 1)
1780C
1790      PRINT, "PLEASE ENTER FILE SIZE IN LLINKS."
1800      READ, LSIZE
1810* - - COMPUTE NUMBER OF SECTORS TO ERASE.
1820      NUMB = (LSIZE * 320)/76
1830      IF(NUMB.GT.40) GO TO 8
1840      PRINT 6, LSIZE, NUMB
1850 6    FORMAT (1X, "A FILE SIZE OF ", I3, " LLINKS YIELDS ONLY",
1860&            I3, " USEABLE SECTORS.",/, "SIMSIZ REQUIRES AT ",
1870&            "LEAST 40 SECTORS TO FUNCTION.")
1880      RETURN
1890C
1900* - - CLEAR OUT FILE SPACE.
1910 8    DO 10 I = 1, NUMB
1920      SECNO = I
1930      CALL WSEC (CARD, SECNO, 0)
1940 10   CONTINUE
1950C
1960      PRINT, "ENTER A 6-CHARACTER NAME FOR THIS DATA BASE."
1970      READ, FNAME
1980      CALL DATIM (TODAY, HRS)
```

```
1990      DATES(1) = TODAY
2000      DATES(2) = TODAY
2010      DATES(3) = TODAY
2020* - - DIRECTORY STARTING SECTOR.
2030      DIR(1) = 2
2040* - - DIRECTORY SECTORS ALLOCATED.
2050      DIR(2) = 3
2060* - - CURRENT NUMBER OF DIRECTORY ENTRIES.
2070      DIR(3) = 1
2080* - - INFO STARTING SECTOR.
2090      INFO(1) = 6
2100* - - NUMBER OF INFO SECTORS.
2110      INFO(2) = 3
2120* - - CURRENT NUMBER OF INFO ENTRIES IN TABLE.
2130      INFO(3) = 7
2140*     STARTING SECTOR OF DATA.
2150      DATAS(1) = 9
2160* - - DATA SECTORS ALLOWED.
2170      DATAS(2) = NUMB - DATAS(1)
2180* - - NEXT AVAILABLE DATA SECTOR.
2190      DATAS(3) = 9
2200C
2210* - - SET NUMBER OF RECORDS CURRENTLY ON FILE.
2220      RNUM = 0
2230* - - CLEAR DIRECTORY SECTION.
2240      DO 100 I = 1, 40
2250      SYSCOD(I) = ' '
2260      SYSNAM(I) = ' '
2270      DSSEC(I) = 0
2280      DESEC(I) = 0
2290      DNSEC(I) = 0
2300      NDEL(I) = 0
2310 100  CONTINUE
2320C
2330* - - LOAD DIRECTORY.
2340      DO 120 I = 1, NDIR
2350      SYSCOD(I) = COD(I)
2360      SYSNAM(I) = NAM(I)
2370      DSSEC(I) = DATAS(3)
2380      DESEC(I) = DATAS(3)
2390      DATAS(3) = DATAS(3) + 1
2400      DNSEC(I) = 1
2410 120  CONTINUE
2420C
2430* - - LOAD COMMON INSEC.
2440      DO 130 I = 1, 15
2450      CPUC(I) = CPUCB(I)
2460      CPUN(I) = CPUNB(I)
2470      CPUS(I) = CPUSB(I)
2480      CPUT(I) = CPUTB(I)
2490      IF(I.GT.10) GO TO 130
2500      LANGC(I) = LANGCB(I)
```

```
2510      LANGN(I) = LANGNB(I)
2520      LANGS(I) = LANGSB(I)
2530 130  CONTINUE
2540C
2550* - - LOAD COMMON INFOS.
2560      DO 140 I = 1, 20
2570      SIMCOD(I) = SIMCODB(I)
2580      SIMNAM(I) = SIMNAMB(I)
2590      SIMFAC(I) = SIMFACB(I)
2600      SIMTXT(I) = SIMTXTB(I)
2610 140  CONTINUE
2620C
2630      DIR(3) = NDIR
2640* - - WRITE ID AND DIRECTORY COMMONS BACK TO DATA BASE.
2650      CALL RWHEAD(2)
2660C
2670      PRINT, "BUILD PROCESSING COMPLETE."
2680      RETURN
2690      END
```

SUBROUTINE ST2AR

```
1000CST2AR    ROUTINE TO CONVERT STRINGS TO ARRAYS.
1010C
1020      SUBROUTINE ST2AR (ANSW, MODE, FC, *, *)
1030C
1040C       MODE:
1050C        1 = READ ALL 3 TYPES OF DATA.
1060C        2 = READ RECORD NUMBERS ONLY.
1070C        3 = READ FIELD NUMBERS ONLY.
1080C        4 = READ FILE VALUES ONLY.
1090C
1100C       FC:
1110C            FILE CODE TO USE FOR READING ADDITIONAL DATA.
1120C
1130C      1ST ALTERNATE RETURN:
1140C            REREAD INPUT DATA.
1150C
1160C      2ND ALTERNATE RETURN:
1170C            EOF ON READ FILE.
1180C
1190      CHARACTER  ANSW*84, HOLD*30, FORM1*5/'(I  )'/, TEMP*1/' '/,
1200&              STR*8(15), LSTR*30, QUOTE*1/1H'/, BLANKS*8/' '/,
1210&              LBLNK*30/" "/, NULL*6/'*NULL*'/
1220C
1230      INTEGER  AP, RP, FP, SP, REC(20), FLD(15), HP, FC
1240C
1250      LOGICAL  RF, FF, SF, QF, DASH, SEMI, DONE, RDR, RDF, RDT,
1260&              EARLY
1270C
1280      COMMON /ARRAYS/ RP, REC, FP, FLD, SP, STR, LSTR, NFLDS
1290C
1300+ - - RESET POINTERS.
1310      NSPACE = 0
1320      AP = 0; HP = 1; QF = .F.; EARLY = .F.
1330      RF = .F.; FF = .F.; SF = .F.; DONE = .F.; SEMI = .F.
1340      DASH = .F.; RDR = .F.; RDF = .F.; RDT = .F.
1350      IF(MODE.GT.4) RETURN
1360      GO TO (1, 2, 3, 4), MODE
1370 1    RF = .T.; GO TO 5
1380 2    RF = .T.; RDR = .T.; GO TO 5
1390 3    FF = .T.; RDF = .T.; GO TO 8
1400 4    SF = .T.; RDT = .T.; GO TO 12
1410 5    DO 7 I = 1, 20
1420 7    REC(I) = 0
1430      RP = 0
1440      IF(RDR) GO TO 20
1450 8    DO 10 I = 1, 15
1460 10   FLD(I) = 0
```

```
1470      FP = 0
1480      IF(RDF) GO TO 20
1490 12   DO 14 I = 1, 15
1500 14   STR(I) = BLANKS
1510      SP = 0
1520      LSTR = NULL
1530 20   AP = AP + 1
1540      IF(AP.LE.84) GO TO 25
1550* - - READ CONTINUATION OF STRING.
1560 21   READ (FC, 22, END=295) ANSW
1570 22   FORMAT (A84)
1580      AP = 1
1590* - - EXTRACT NEXT CHARACTER.
1600 25   CALL CONCAT (TEMP, 1, ANSW, AP, 1)
1610      IF(TEMP.EQ.',') GO TO 200
1620      IF(TEMP.EQ.'-') GO TO 280
1630      IF(TEMP.EQ.';') GO TO 300
1640      IF(TEMP.EQ.'$') GO TO 380
1650      IF(RF .OR. FF) GO TO 400
1660* - - MUST BE IN STRING SECTION.
1670      IF(TEMP.EQ.QUOTE) GO TO 100
1680      IF(TEMP.EQ.'"') GO TO 100
1690      IF(TEMP.EQ.' ' .AND. (.NOT. QF)) GO TO 140
1700 85   CALL CONCAT (HOLD, HP, TEMP, 1, 1)
1710      HP = HP + 1
1720      IF(HP.LT.31) GO TO 20
1730      EARLY = .T.
1740* - - HAVE REACHED END OF HOLD VARIABLE.  MOVE IT TO LSTR.
1750 90   LSTR = HOLD
1760      SP = SP + 1
1770 92   HOLD = LBLNK; HP = 1
1780      NSPACE = 0
1790      IF(SEMI) GO TO 340
1800      IF(DONE) GO TO 95
1810      GO TO 20
1820 95   IF(NFLDS.EQ.0) GO TO 97
1830      IF(SP.EQ.NFLDS) RETURN
1840      GO TO 99
1850 97   IF(SP.EQ.FP) RETURN
1860 99   DONE = .F.
1870      GO TO 21
1880C
1890* - - HAVE FOUND A QUOTE.
1900 100  IF(QF) GO TO 110
1910* - - BEGINNING OF QUOTE.
1920      QF = .T.; GO TO 20
1930* - - END OF QUOTE.
1940 110  QF = .F.
1950* - - CHECK FOR PREMATURE TERMINATION OF STRING AT 30 CHAR.
1960      IF(.NOT. EARLY) GO TO 115
1970      EARLY = .F.; GO TO 92
1980* - - CHECK FOR LONG FIELD STRING (FLD 15).
```

```
1990 115  IF(HP.GT.9) GO TO 90
2000* - - PLACE HOLD STRING IN NEXT AVAILABLE SLOT.
2010 120  IF(HP.LT.2) GO TO 92
2020      IF(.NOT. EARLY) GO TO 125
2030      EARLY = .F.; GO TO 92
2040 125  SP = SP + 1
2050      IF(SP.LE.15) GO TO 130
2060      PRINT, "TOO MANY STRING VARIABLES.  RE-INPUT DATA."
2070      RETURN 1
2080 130  CALL CONCAT (STR(SP), 1, HOLD, 1, HP - 1)
2090      GO TO 92
2100 140  NSPACE = NSPACE + 1
2110      IF(NSPACE.LE.3) GO TO 20
2120      DONE = .T.
2130      IF(RF .OR. FF) RETURN
2140      GO TO 115
2150C
2160* - - HAVE FOUND COMMA.
2170 200  IF(RF) GO TO 220
2180      IF(FF) GO TO 220
2190      IF(QF) GO TO 85
2200* - - MUST BE SEPERATING STRING VALUES.
2210      IF(HP.GT.1) GO TO 120
2220* - - IGNORE IT.
2230      GO TO 20
2240* - - DECODE AND STORE.
2250 220  IF(HP.GE.2) GO TO 222
2260      PRINT, "CONSECUTIVE COMMAS IN 'RECORD' OR 'FIELD' SECTION."
2270      PRINT, "IGNORE SECOND CHARACTER."
2280      GO TO 20
2290 222  ENCODE (FORM1, 225) HP - 1
2300 225  FORMAT (T3, I2)
2310      IF(FF) GO TO 250
2320      RP = RP + 1
2330      IF(RP.LE.20) GO TO 230
2340      PRINT, "TOO MANY RECORD VARIABLES.  OVERWRITING LAST VARIABLE."
2350      RP = 20
2360 230  DECODE (HOLD, FORM1) REC(RP)
2370      IF(DASH) GO TO 235
2380      IF(TEMP.EQ.'-') DASH = .T.
2390      GO TO 92
2400 235  REC(RP) = -REC(RP)
2410      DASH = .F.
2420      GO TO 92
2430 250  FP = FP + 1
2440      IF(FP.LE.15) GO TO 260
2450      PRINT, "TOO MANY FIELD VARIABLES.  OVERWRITING LAST VARIABLE."
2460      FP = 15
2470 260  DECODE (HOLD, FORM1) FLD(FP)
2480      GO TO 92
2490C
2500* - - HAVE FOUND DASH.
```

```
2510 280  IF(RF) GO TO 220
2520      IF(.NOT. FF) GO TO 85
2530      PRINT, "THE '-' IS NOT PERMITTED IN THE FIELD DESIGNATOR STRING."
2540      PRINT, "EACH FIELD TO BE MODIFIED MUST BE IDENTIFIED SEPERATELY."
2550 290  RETURN 1
2560C
2570 295  RETURN 2
2580C
2590* - - HAVE FOUND ';'.
2600 300  IF( QF) GO TO 85
2610      SEMI = .T.
2620      IF(RF .OR. FF) GO TO 222
2630      GO TO 20
2640 340  IF(FF) GO TO 345
2650      IF(RF) RF = .F.
2660      FF = .T.
2670      GO TO 350
2680 345  FF = .F.; SF = .T.
2690 350  SEMI = .F.
2700      GO TO 20
2710C
2720* - - HAVE FOUND '#'.  END OF INPUT STRING.
2730 380  IF(QF) GO TO 85
2740      DONE = .T.
2750      IF(RF .AND. HP.GE.2) GO TO 222
2760      IF(SF .AND. HP.GE.2) GO TO 120
2770      IF(.NOT. FF) RETURN
2780      PRINT, "PREMATURE TERMINATION MARK IN FIELD DESIGNATOR STRING."
2790      RETURN 1
2800C
2810* - - CHECK TO SEE IF CHARACTER IS A LEGITIMATE NUMBER.
2820 400  IF(TEMP.EQ.'0' .OR. TEMP.EQ.'1' .OR. TEMP.EQ.'2' .OR.
2830&        TEMP.EQ.'3' .OR. TEMP.EQ.'4' .OR. TEMP.EQ.'5' .OR.
2840&        TEMP.EQ.'6' .OR. TEMP.EQ.'7' .OR. TEMP.EQ.'8' .OR.
2850&        TEMP.EQ.'9') GO TO 85
2860      IF(TEMP.EQ.' ' .AND. HP.GE.2) GO TO 222
2870      IF(TEMP.EQ.' ' .AND. (RDR .OR. RDF)) GO TO 140
2880      GO TO 20
2890      END
```

SUBROUTINE TMOD

```
1000CTMOD   TEMP FILE MODIFIER ROUTINE.
1010C
1020        SUBROUTINE TMOD
1030C
1040        CHARACTER  ANSW*84, LSTR*30, STR*8(15), ANS*1, BUFF*450,
1050&                  SREC*90(5), TEMP*1
1060C
1070        INTEGER  SECNO, FC, RP, FP, SP, REC(200), FLD(15), RN
1080C
1090        LOGICAL  DIREUP, INFOUP, DATAUP, WINFO, SERIES
1100C
1110        EQUIVALENCE  (BUFF, SREC(1))
1120C
1130        COMMON /MODBUF/ BUFF
1140C
1150        COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
1160C
1170        COMMON /ARRAYS/ RP, REC, FP, FLD, SP, STR, LSTR, NFLDS
1180C
1190C
1200        IF(NSEL.GT.0) GO TO 20
1210        PRINT, "THERE ARE NO RECORDS ON THE TEMP SELECT FILE."
1220        RETURN
1230 20     SERIES = .F.
1240        NFLDS = 0
1250        IF(.NOT. WINFO) GO TO 40
1260        PRINT, "INSTRUCTIONS [THEY ARE LONG] (Y OR N)?"
1270        READ, ANS
1280        IF(ANS.EQ.'Y') GO TO 25
1290        WINFO = .F.; GO TO 40
1300 25     WINFO = .T.
1310        PRINT, "THIS ROUTINE WILL MODIFY (OR DELETE MARK) ONE OR MORE"
1320        PRINT, "RECORDS WHICH HAVE BEEN PLACED ON THE 'TEMP' "
1330        PRINT, "FILE.  IF YOU HAVE NOT ALREADY PLACED RECORDS ON THE"
1340        PRINT, "TEMP FILE, DROP BACK TO THE 'FUNCTION?' LEVEL AND CHOOSE"
1350        PRINT, "THE SELECT (S) FUNCTION.  "
1360        PRINT, " "
1370        PRINT, "RECORDS ON THE TEMP FILE MAY BE MODIFIED (OR DELETED)"
1380        PRINT, "INDIVIDUALLY OR IN GROUPS OF CONTIGUOUS RECORDS."
1390        PRINT, "THE RECORDS TO BE MODIFIED (OR DELETED) ARE DESIGNATED"
1400        PRINT, "WHEN YOU ENTER A CONTROL STRING OF THE RECORD NUMBERS"
1410        PRINT, "YOU WISH TO MODIFY (DELETE), I.E., ENTERING THE STRING"
1420        PRINT, "   1, 4, 8, 76, 100-123, 127 "
1430        PRINT, "WOULD INDICATE THAT YOU WANT TO MODIFY RECORDS 1, 4, 8,"
1440        PRINT, "76, 100 THRU 123, AND 127 ON THE TEMP FILE.  A DASH"
1450        PRINT, "BETWEEN NUMBERS MEANS 'INCLUSIVE' AND WILL CAUSE ALL"
1460        PRINT, "RECORDS, STARTING AT THE FIRST RECORD NUMBER AND ENDING"
```

```
1470      PRINT, "WITH THE SECOND NUMBER, TO BE CHOSEN FOR MODIFICATION"
1480      PRINT, "(DELETION)."
1490      PRINT, " "
1500      PRINT, "THE FIELDS (WITHIN RECORDS) TO BE MODIFIED ARE CHOSEN"
1510      PRINT, "BY ENTERING THE NUMBERS OF THE DESIRED FIELDS.  THE FIELD"
1520      PRINT, "VALUES RANGE FROM 1 TO 18 AND ARE THE SAME FIELDS"
1530      PRINT, "DESCRIBED IN THE ADD ROUTINE INSTRUCTIONS PLUS THREE"
1540      PRINT, "'FLAG' FIELDS."
1550      PRINT, "NOTE:  IF THE 'FUNCTION CODE' (FIELD 2) FOR A RECORD"
1560      PRINT, "HAS BEEN MODIFIED, IT WILL NOT BE POSSIBLE TO UPDATE"
1570      PRINT, "THE OLD RECORD.  THE ORIGINAL (UN-MODIFIED) RECORD(S)"
1580      PRINT, "MUST FIRST BE DELETED FROM THE DATA BASE.  THE DELETE"
1590      PRINT, "MARKED RECORDS ON THE TEMP FILE ARE THEN MODIFIED BY"
1600      PRINT, "CHANGING THE FUNCTION CODE TO THE NEW VALUE.  THEN THE"
1610      PRINT, "MODIFIED RECORD(S) MUST BE ADDED TO THE DATA BASE "
1620      PRINT, "USING THE SIMSIZ ADD FUNCTION."
1630      PRINT, " "
1640      PRINT, "DO YOU WANT A LISTING OF THE FIELD NUMBERS, FORMATS, "
1650      PRINT, "AND CONTENTS (Y OR N)?"
1660      READ, ANS
1670      IF(ANS.EQ.'N') GO TO 35
1680 30   PRINT, "THE CONTENTS OF RECORD FIELDS 1 THRU 18"
1690      PRINT, "ARE AS FOLLOWS:"
1700      PRINT, "FIELD  FIELD"
1710      PRINT, "NUMBER FORMAT  DISCRIPTION"
1720      PRINT, "  1     A1    SIMULATOR CODE"
1730      PRINT, "  2     A2    FUNCTION CODE"
1740      PRINT, "  3     A2    SUBSYSTEM CODE"
1750      PRINT, "  4     A8    MFG MODULE CODE"
1760      PRINT, "  5     A1    PROGRAM LANGUAGE"
1770      PRINT, "  6     F6.2  TIMING - MFG EST"
1780      PRINT, "  7     I5    INSTR - MFG EST"
1790      PRINT, "  8     I5    DATA - MFG EST"
1800      PRINT, "  9     F5.1  CPS - CURRENT"
1810      PRINT, " 10     F6.2  TIMING - CURRENT"
1820      PRINT, " 11     I5    INSTR - CURRENT"
1830      PRINT, " 12     I5    DATA - CURRENT"
1840      PRINT, " 13     I2    YEAR OF UPDATE"
1850      PRINT, " 14     A2    COMPUTER SYSTEM"
1860      PRINT, " 15     A30   COMMENTS TEXT"
1870      PRINT, " 16     A1    USER FLAG 1"
1880      PRINT, " 17     A1    USER FLAG 2"
1890      PRINT, " 18     A1    USER FLAG 3"
1900      IF(ANS.EQ.'F') GO TO 40
1910 35   PRINT, " "
1920      PRINT, "FOR THE DELETE FUNCTION, ONLY THE RECORD NUMBER(S) TO BE"
1930      PRINT, "DELETED NEED BE SPECIFIED.  FOR THE MODIFY FUNCTION, YOU"
1940      PRINT, "MUST ENTER THE RECORD NUMBER(S); FIELD NUMBER(S); AND"
1950      PRINT, "NEW FIELD VALUES.  FOR EXAMPLE:"
1960      PRINT, "  RN1, RN2, RN3; FN1, FN2, FN3; NFLD1, NFLD2, NFLD3"
1970      PRINT, "IS A GENERALIZED INPUT CONTROL STRING WHERE:"
1980      PRINT, "  - RN1, RN2, AND RN3 REPRESENT THE NUMBERS OF THE "
```

```
1990      PRINT, "    RECORDS TO BE MODIFIED"
2000      PRINT, "  - FN1, FN2, AND FN3 ARE THE NUMBERS OF THE FIELDS TO"
2010      PRINT, "    BE MODIFIED, IN 'EACH' RECORD."
2020      PRINT, "  - NFLD1, NFLD2, AND NFLD3 ARE CHARACTER STRINGS WHICH"
2030      PRINT, "    WILL REPLACE THE RECORD FIELDS SPECIFIED  BY THE "
2040      PRINT, "    'FN' NUMBERS."
2050      PRINT, "NOTE:  IF THE CHARACTER STRINGS CONTAIN EMBEDDED"
2060      PRINT, "BLANKS OR SPECIAL CHARACTERS (, ; OR -) THE STRINGS MUST"
2070      PRINT, "BE ENCLOSED IN QUOTES, I.E., "
2080      PRINT, " 2, 7, 100-110; 3, 4, 11, 15; SD, 'MOD-21', 2478, 'HAMS 2'"
2090      PRINT, " "
2100      PRINT, "A SEMI-COLON IS USED TO SEPERATE THE RECORD NUMBERS "
2110      PRINT, "FROM FIELD NUMBERS, AND ALSO FIELD NUMBERS FROM FIELD"
2120      PRINT, "VALUES."
2130C
2140 40   PRINT, "ENTER D (DELE), M (MOD), F (FORMAT), E (END), L (LIST),",
2150&      " OR I(INSTR)."
2160      READ, ANS
2170      IF(ANS.EQ.'M') GO TO 70
2180      IF(ANS.EQ.'F') GO TO 30
2190      IF(ANS.EQ.'I') GO TO 25
2200      IF(ANS.EQ.'D') GO TO 42
2210      IF(ANS.EQ.'L') GO TO 190
2220      IF(ANS.EQ.'E') RETURN
2230      GO TO 40
2240* - - DELETE RECORDS.
2250 42   MODE = 2
2260      PRINT, "ENTER THE NUMBERS OF RECORDS TO BE DELETE MARKED, PREFERABLY"
2270      PRINT, "IN ASCENDING ORDER."
2280 43   READ 45, ANSW
2290 45   FORMAT (A84)
2300      CALL ST2AR (ANSW, 2, 85, $50, $800)
2310      GO TO 60
2320 50   PRINT, "PLEASE RE-ENTER THE RECORD NUMBERS."
2330      GO TO 43
2340 60   CALL DARAY (MODE)
2350      PRINT, "DO YOU WANT TO CHANGE THESE NUMBERS (Y OR N)?"
2360      READ, ANS
2370      IF(ANS.EQ.'Y') GO TO 50
2380      GO TO 100
2390C
2400* - - MODIFY RECORDS.
2410 70   MODE = 1
2420      PRINT, "ENTER THE RECORD NUMBERS; FIELD NUMBERS; FIELD VALUES."
2430 75   READ 45, ANSW
2440      CALL ST2AR (ANSW, 1, 85, $80, $800)
2450* - - CHECK FOR VALID FIELD NUMBERS.
2460      DO 78 I = 1, FP
2470      IF(FLD(I).GT.0 .AND. FLD(I).LE.18) GO TO 78
2480      PRINT 76, FLD(I)
2490 76   FORMAT (1X,"FIELD NUMBER",I3," IS OUTSIDE THE LIMITS (1-18).")
2500      GO TO 80
```

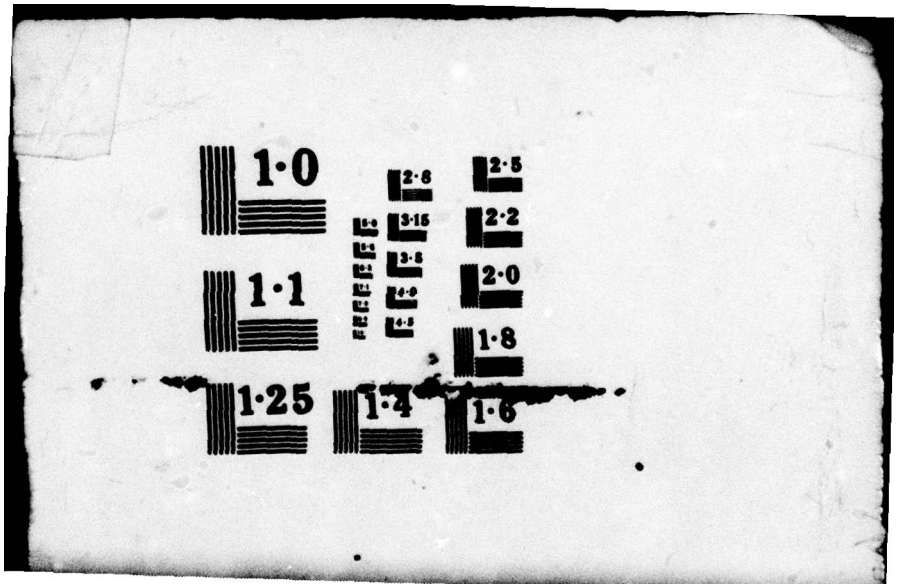
```
2510 78   CONTINUE
2520      GO TO 90
2530 80   PRINT, "PLEASE RE-ENTER THE ENTIRE CONTROL STRING."
2540      GO TO 75
2550 90   CALL DARAY (MODE)
2560      PRINT, "DO YOU WANT TO CHANGE THESE VALUES (Y OR N)?"
2570      READ, ANS
2580      IF(ANS.EQ.'Y') GO TO 80
2590C
2600* - - SELECT RECORDS.
2610 100  IP = 1
2620 105  RN = REC(IP)
2630      IF(RN.LE.NSEL) GO TO 108
2640 106  PRINT 107, RN, NSEL
2650 107  FORMAT (1X, "YOU HAVE SELECTED RECORD NUMBER ", I4,
2660&     ".  THERE ARE ONLY ", I4, " RECORDS ",/,"ON THE TEMP FILE.")
2670      GO TO 80
2680 108  IF(RN.GT.0) GO TO 110
2690      IP = IP + 1; GO TO 105
2700* - - COMPUTE SECTOR CONTAINING RECORD.
2710 110  IREM = MOD(RN,5)
2720      SECNO = RN / 5
2730      IF(IREM.GT.0) SECNO = SECNO + 1
2740      LREC = SECNO * 5
2750      IF(IREM.EQ.0) IREM = 5
2760* - - READ SECTOR FROM FILE INTO BUFF.
2770      READ (20'SECNO, END=800, ERR=850) BUFF
2780* - - MARK THE RECORD AS EITHER DELETED (D) OR MODIFIED (M).
2790 120  TEMP = 'D'
2800      IF(MODE.EQ.1) TEMP = 'M'
2810      CALL CONCAT (SREC(IREM), 2, TEMP, 1, 1)
2820      IF(MODE.EQ.2) GO TO 140
2830* - - MODIFY THE RECORD FIELDS.
2840      CALL RECMOD (IREM)
2850* - - CHECK THE NEXT RECORD TO BE MODIFIED.
2860 140  IF(SERIES) GO TO 170
2870      IP = IP + 1
2880* - - CHECK TO SEE IF WE HAVE PROCESSED ALL THE LIST.
2890      IF(IP.GT.RP) GO TO 180
2900* - - EXTRACT THE NEXT RECORD NUMBER, BUT DON'T WIPE OLD RN YET.
2910      IR = REC(IP)
2920* - - CHECK TO SEE IF NUMBERS ARE OUT OF ORDER.
2930      IF(IR.GT.0 .AND. IR.LT.RN) GO TO 180
2940      IF(IR.GT.0) GO TO 150
2950* - - A NEGATIVE RECORD NUMBER; MEANS DO ALL FROM HERE TO IR.
2960      IR = -IR
2970* - - CHECK TO SEE THAT IR IS NOT TOO LARGE.
2980      IF(IR.LE.NSEL) GO TO 145
2990      RN = IR
3000      GO TO 106
3010* - - CHECK TO MAKE SURE IR IS GREATER THAN RN.  IF NOT, SKIP TO NEXT REC.
3020 145  IF(IR.LE.RN) GO TO 140
```

AD-A073 015 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL--ETC F/G 9/2
A PARAMETRIC MANAGEMENT TOOL FOR ESTIMATING SIMULATOR SOFTWARE --ETC(U)
JUN 79 G N FREY, K L WILDUNG

UNCLASSIFIED AFIT-LSSR-4-79A NL

3 OF 3
AD
A073015

END
DATE FILMED
9 79
DDC


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15

```
3030      SERIES = .T.
3040      RN = RN + 1
3050      GO TO 160
3060 150  RN = IR
3070      IF(RN.LE.NSEL) GO TO 160
3080      GO TO 106
3090* - - CHECK TO SEE IF NEXT RECORD IS IN THIS SECTOR.
3100 160  IF(RN.LE.LREC) GO TO 165
3110* - - WRITE MODIFIED SECTOR BACK TO TEMP FILE.
3120      WRITE (20'SECNO, ERR=825) BUFF
3130      GO TO 110
3140* - - IT IS IN THIS SECTOR
3150 165  IREM = MOD(RN, 5)
3160      IF(IREM.EQ.0) IREM = 5
3170      GO TO 120
3180* - - INCREMENT RN AND SEE IF WE HAVE REACHED IR YET.
3190 170  RN = RN + 1
3200      IF(RN.LE.IR) GO TO 160
3210* - - HAVE COMPLETED THAT SERIES.  RESET.
3220      SERIES = .F.; GO TO 140
3230* - - WRITE LAST SECTOR BACK TO TEMP FILE.
3240 180  WRITE (20'SECNO, ERR=825) BUFF
3250      IF(IP.LE.RP)GO TO 105
3260* - - HAVE FINISHED MODIFYING/DELETING RECORDS.
3270      PRINT, "MODIFY (DELETE) ACTION COMPLETE."
3280      GO TO 40
3290* - - CALL LISTR SUBROUTINE.  SET FUNCTION CODE FOR 1.
3300 190  FC = 1
3310      CALL LISTR (FC)
3320      GO TO 40
3330C
3340 800  PRINT, "EOF ON TEMP FILE READ."
3350      GO TO 860
3360 825  PRINT, "ERROR RETURN WHILE WRITING TO TEMP FILE."
3370      GO TO 860
3380 850  PRINT, "ERROR RETURN ON TEMP FILE READ."
3390 860  PRINT, "TERMINATING FUNCTION."
3400      RETURN
3410      END
```

SUBROUTINE UPDATE

```
1000CUPDATE   WRITE MODIFIED/DELETED RECORDS FROM TEMP TO DB.
1010C
1020      SUBROUTINE UPDATE
1030C
1040      CHARACTER  SYSCOD*2(40), SYSNAM*12(40), ANS*1, CARD*450,
1050&                CREC*90(5), SC*2, OSC*2, TEMP*1, SF*1,
1060&                BUFF*450, BREC*90(5), FNAME*6, DATES*8(3)
1070C
1080      INTEGER  SECNO, TSEC, CONT, DSSEC(40), DESEC(40),
1090&              DNSEC(40), NDEL(40), CP, BP, RNUM, ORNUM, DIR(3),
1100&              INFO(3), DATAS(3), BASER, TREC
1110C
1120      LOGICAL  DIREUP, INFOUP, DATAUP, WINFO, CHANCE, FIRST
1130C
1140      EQUIVALENCE  (CARD, CREC(1)), (BUFF, BREC(1))
1150C
1160      COMMON /MISC/ DIREUP, INFOUP, DATAUP, WINFO, NSEL
1170C
1180      COMMON /IDSEC/ FNAME, DATES, LSIZE, DIR, INFO, DATAS, RNUM
1190C
1200      COMMON /DRSEC/ SYSCOD, SYSNAM, DSSEC, DESEC, DNSEC, NDEL
1210C
1220C
1230      IF(NSEL.GT.0) GO TO 20
1240      PRINT, "THERE ARE NO RECORDS ON THE TEMP FILE."
1250      RETURN
1260 20   IF(.NOT. WINFO) GO TO 40
1270      PRINT, "INSTRUCTIONS (Y OR N)?"
1280      READ, ANS
1290      IF(ANS.EQ.'Y') GO TO 25
1300      WINFO = .F.
1310      GO TO 40
1320 25   WINFO = .T.
1330      PRINT, "THIS ROUTINE WILL UPDATE THE DATA BASE WITH RECORDS"
1340      PRINT, "FROM THE PREVIOUSLY CREATED 'TEMP' FILE.  "
1350      PRINT, " - RECORDS THAT HAVE BEEN MODIFIED WILL OVER-WRITE"
1360      PRINT, "   THE ORIGINAL RECORD IN THE DATA BASE."
1370      PRINT, " - RECORDS THAT HAVE BEEN DELETE MARKED WILL BE DELETE"
1380      PRINT, "   MARKED IN THE DATA BASE."
1390      PRINT, " - RECORDS ON THE TEMP FILE WHICH HAVE NOT BEEN "
1400      PRINT, "   MODIFIED OR DELETE MARKED WILL BE IGNORED."
1410      PRINT, " "
1420      PRINT, "NOTE:  IF THE 'FUNCTION CODE' (FIELD 2) FOR A RECORD"
1430      PRINT, "HAS BEEN MODIFIED, IT WILL NOT BE POSSIBLE TO UPDATE"
1440      PRINT, "THE OLD RECORD.  THE ORIGINAL (UN-MODIFIED) RECORD(S)"
1450      PRINT, "MUST FIRST BE DELETED FROM THE DATA BASE.  THE DELETE"
1460      PRINT, "MARKED RECORDS ON THE TEMP FILE ARE THEN MODIFIED BY"
```

```
1470      PRINT, "CHANGING THE FUNCTION CODE TO THE NEW VALUE.  THEN THE"
1480      PRINT, "MODIFIED RECORD(S) MUST BE ADDED TO THE DATA BASE "
1490      PRINT, "USING THE SIMSIZ ADD FUNCTION."
1500      PRINT, " "
1510      PRINT, "ALL RECORDS ON THE TEMP FILE REMAIN ON THE TEMP FILE"
1520      PRINT, "AND ARE UNCHANGED, ONLY THE RECORDS IN THE DATA BASE ARE"
1530      PRINT, "ALTERED BY THIS ROUTINE."
1540C
1550* - - DETERMINE HOW MANY SECTORS OF THE TEMP FILE HOLD DATA.
1560 40   IREM = MOD(NSEL, 5)
1570      NUMSEC = NSEL / 5
1580      IF(IREM.GT.0) NUMSEC = NUMSEC + 1
1590* - - RESET FLAGS AND COUNTERS.
1600 50   TSEC = 1; SC = ' '; OSC = ' '; FIRST = .T.; BASER = -5
1610      SF = ' '; ORNUM = 0; CHANGE = .F.; TEMP = ' '
1620* - - GET THE NEIT SECTOR OF RECORDS FROM THE TEMP FILE.
1630 60   READ (20'TSEC, END=220, ERR=850) BUFF
1640      ISTOP = 5
1650      BASER = BASER + 5
1660      NLEFT = NSEL - BASER
1670      IF(NLEFT.LT.5) ISTOP = NLEFT
1680      DO 200 BP = 1, ISTOP
1690* - - EXTRACT STATUS FLAG FROM RECORD.
1700      CALL CONCAT (TEMP, 1, BREC(BP), 2, 1)
1710      IF(TEMP.NE.'D' .AND. TEMP.NE.'M') GO TO 200
1720* - - EXTRACT SYSTEM CODE FROM RECORD.
1730      CALL CONCAT (SC, 1, BREC(BP), 3, 2)
1740* - - IS IT THE SAME CODE AS LAST RECORD?
1750      IF(SC.EQ.OSC) GO TO 110
1760      OSC = SC
1770* - - NO, WRITE CARD BUFFER BACK TO DB.
1780*     UNLESS THIS IS THE FIRST RECORD TO BE PROCESSED.
1790      IF(FIRST) GO TO 80
1800      IF(.NOT. CHANGE) GO TO 80
1810      CALL WSEC (CARD, SECNO, CONT)
1820      CHANGE = .F.
1830* - - CHECK DIRECTORY FOR THIS SYSTEM CODE.
1840 80   DO 90 I = 1, DIR(3)
1850      IF(SC.EQ.SYSCOD(I)) GO TO 110
1860 90   CONTINUE
1870      PRINT 100, SC
1880 100  FORMAT (1X, "NO DIRECTORY ENTRY FOR FUNCTION CODE ", A2, ".",/,
1890&     "THE FOLLOWING RECORD WILL NOT BE PLACED IN THE DATA BASE:")
1900 105  ITOTAL = 0
1910      K = BP
1920      CALL DSPLAY (BUFF, 1, ITOTAL, BASER + K, K)
1930      GO TO 200
1940* - - SET STARTING SECTOR VALUE.
1950 110  SECNO = DSSEC(I)
1960      ORNUM = 0
1970* - - READ DATA BASE.
1980 115  CALL RSEC (CARD, SECNO, CONT, *800)
```

```
1990      FIRST = .F.
2000*  - - CHECK FOR MATCH OF DB RECORD NUMBERS.
2010*  - - EXTRACT RECORD NUMBER FROM TEMP RECORD.
2020 110  DECODE (BREC(BP), 120) TREC
2030 120  FORMAT (T87, I4)
2040      IF(TREC.GE.ORNUM) GO TO 125
2050*  - - WRITE SECTOR BACK TO DB.
2060      IF(.NOT. CHANGE) GO TO 110
2070      CALL WSEC (CARD, SECNO, CONT)
2080      CHANGE = .F.; GO TO 110
2090*  - - LOOP THROUGH CARD BUFFER LOOKING FOR A MATCH.
2100 125  DO 130 CP = 1, 5
2110*  - - EXTRACT STATUS FLAG.
2120      CALL CONCAT (SF, 1, CREC(CP), 2, 1)
2130*  - - IF NOT 'S' OR 'C' (I.E., MUST BE 'D' OR ' ') SKIP IT.
2140      IF(SF.NE.'S' .AND. SF.NE.'C') GO TO 130
2150      DECODE (CREC(CP), 120) ORNUM
2160      IF(ORNUM.EQ.TREC) GO TO 150
2170 130  CONTINUE
2180*  - - NO MATCH YET.  CHECK TO SEE IF THIS IS LAST SECTOR
2190*     FOR THIS FUNCTION CODE.
2200*  - - FIRST, WRITE THIS SECTOR BACK TO DB.
2210      IF(.NOT. CHANGE) GO TO 135
2220      CALL WSEC (CARD, SECNO, CONT)
2230      CHANGE = .F.
2240 135  IF(SECNO.EQ.DESEC(I)) GO TO 140
2250*  - - UPDATE SECTOR NUMBER AND READ NEXT SECTOR.
2260      SECNO = CONT
2270      CALL RSEC (CARD, SECNO, CONT, $800)
2280      GO TO 125
2290*  - - NO MATCH FOR THIS RECORD.
2300 140  PRINT, "COULD NOT MATCH THE FOLLOWING RECORD.  SKIP IT AND",
2310&     " CONTINUE."
2320      GO TO 105
2330C
2340*  - - HAVE FOUND A MATCH.  ARE WE DELETING OR UPDATING?
2350 150  CHANGE = .T.
2360      IF(TEMP.EQ.'N') GO TO 160
2370*  - - DELETE MARK DB RECORD STATUS FLAG.
2380      CALL CONCAT (CREC(CP), 2, TEMP, 1, 1)
2390*  - - ADD ONE TO DELETE COUNT IN DIRECTORY.
2400      NDEL(I) = NDEL(I) + 1
2410      DIREUP = .T.
2420      GO TO 200
2430C
2440*  - - UPDATE THE DB RECORD WITH THE TEMP RECORD.
2450 160  CREC(CP) = BREC(BP)
2460*  - - PUT STATUS FLAG BACK THE WAY IT WAS.
2470      CALL CONCAT (CREC(CP), 2, SF, 1, 1)
2480 200  CONTINUE
2490C
2500*  - - FINISHED WITH THIS SECTOR.  ARE THERE MORE SECTORS TO PROCESS?
```

```
2510      DATAUP = .T.
2520* - - CHECK TO SEE IF WE ALSO FINISHED THE CARD BUFFER.
2530      IF(CP.LT.5) GO TO 210
2540      IF(.NOT. CHANGE) GO TO 210
2550      CALL WSEC (CARD, SECNO, CONT)
2560      IF(CONT.EQ.0) GO TO 210
2570      SECNO = CONT
2580      CALL RSEC (CARD, SECNO, CONT, $800)
2590      CHANGE = .F.
2600 210  TSEC = TSEC + 1
2610      IF(TSEC.LE.NUMSEC) GO TO 60
2620* - - HAVE FINISHED PROCESSING TEMP RECORDS.
2630      IF(.NOT. CHANGE) GO TO 220
2640* - - WRITE LAST SECTOR BACK TO DB.
2650      CALL WSEC (CARD, SECNO, CONT)
2660 220  PRINT, "DATA BASE UPDATE COMPLETE."
2670      CALL RWHEAD (2)
2680      RETURN
2690C
2700 800  PRINT, "ERROR RETURN WHILE WRITING THE FOLLOWING SECTOR TO",
2710&     " THE DATA BASE:"
2720      ITOTAL = 0
2730      CALL DSPLAY (CARD, CP, ITOTAL, 1, 1)
2740 810  PRINT, "TERMINATING UPDATE RUN."
2750      RETURN
2760 850  PRINT, "ERROR RETURN FROM TEMP FILE READ."
2770      GO TO 810
2780      END
```

SELECTED BIBLIOGRAPHY

A. REFERENCES CITED

1. Aeronautical Systems Division, Air Force Systems Command. Air Force Master-Plan: Simulators for Aircrew Training. Wright-Patterson AFB OH, September 1975.

2. Applied Physics Laboratory, John Hopkins University. DOD Weapons System Software Management Study. June 1975. AD A022 160.

3. Babel, Philip. Technical Specialist for Simulator Software, Simulator Division, Directorate of Equipment Engineering, Aeronautical Systems Division, AFSC, Wright-Patterson AFB OH. Personal interview. 13 October 1978.

4. Brown, Robert W. Deputy Program Manager for Logistics, Integrated Logistics Support Division, Simulator System Program Office, Air Force Acquisition Logistics Division, AFLC, Wright-Patterson AFB OH. Telephone interview. 8 May 1979.

5. Butler, David A. Computational Systems Engineer, Simulator Computer Branch, Aeronautical Systems Division, AFSC, Wright-Patterson AFB OH. Personal interview. 8 December 1978.

6. Cameron, Robert G. Chief Simulator Computer Branch, Simulator Division, Directorate of Equipment Engineering, Aeronautical Systems Division, AFSC, Wright-Patterson AFB OH. Personal interview. 17 October 1978.

7. Colbert, Douglas A. Data Processing Concepts. New York: McGraw-Hill Book Company, 1968.

8. Davis, C. G., and C. R. Vick. "The Software Development System," IEEE Transactions on Software Engineering, January 1977, pp.69-84.

9. Davis, Ruth M. "Reducing Software Management Risks," Defense Systems Management Review, Summer 1978, pp. 16-23.

10. Department of the Air Force. Automatic Data Processing Procedures. AFLCR 171-36. 11 November 1977.

11. Dillen, Second Lieutenant David A. Computational Systems Engineer, Simulator Computer Branch, Aeronautical Systems Division, AFSC, Wright-Patterson AFB OH. Personal interview. 8 May 1979.

12. Martin, James A. Computer Data-Base Organization. Englewood Cliffs NJ: Prentice-Hall, 1975.

13. Meyers, Ware, ed. "The Need for Software Engineering," Computer, February 1978, pp. 12-24.

14. Patterson, Clifford E. Computational Systems Engineer, Simulator Computer Branch, Aeronautical Systems Division, AFSC, Wright-Patterson AFB OH. Personal interview. 9 January 1979.

15. Pieroway, Major Chesley, USAF. Chief, Program Control Branch, Simulator Systems Project Office, Aeronautical Systems Division, AFSC, Wright-Patterson AFB OH. Telephone interview. 19 October 1978.

16. Ross, Captain Milton C., USAF, and Captain Jerald L. Yeager. "A Parametric Costing Model for Flight Simulator Acquisition." Unpublished master's thesis. LSSR 22-76B, AFIT/SL, Wright-Patterson AFB OH, September 1976. AD A032 326.

17. Stucki, L. G. "New Direction in Automated Tools for Improving Software Quality," in Current Trends in Programming Methodology, Vol. 2, Program Validation. Englewood Cliffs NJ: Prentice-Hall, 1977.

18. U.S. General Accounting Office. Department of Defense Use of Flight Simulators. B-157905. Washington: Government Printing Office, June 1975.

19. Wasserman, Anthony I., and L. A. Belady. "Software Engineering: The Turning Point," Computer, September 1978, pp. 30-41.

20. Zaluski, Joseph J. Chief, Budget Control Branch, Simulator Systems Program Office, Aeronautical Systems Division, AFSC, Wright-Patterson AFB OH. Personal interview. 6 October 1978.

B. RELATED SOURCES

Arceneaux, Major Ronald J., USAF, and Captain George E. Farschman, Jr., USAF. "An Assessment of Relevant Decision Making Factors for Organic Versus Contract Maintenance Options on USAF Flight Simulators." Unpublished master's thesis. LSSR 7-77B, AFIT/SL, Wright-Patterson AFB OH, September 1977. AD A047136.

Department of the Air Force. Acquisition and Support Procedures for Computer Resources in Systems. AFR 800-14. 26 September 1975.

Devenny, Captain Thomas J., USAF. "An Exploratory Study of Software Cost Estimating at the Electronic Systems Division." Unpublished master's thesis. GSM/SM/76S-4, Wright-Patterson AFB OH, July 1976. AD 030162.

General Research Corporation. Cost Reporting Elements and Activity Cost Tradeoffs for Defense System Software (Executive Summary). May 1977. AD A053021

Nelson, Eldred. "Developing a Software Cost Methodology," Compcon 76, Fall 1976, pp. 144-145.

Premo, A. F., Jr. "Computer Software: Estimating Guidelines," Compcon 76, pp. 146-151.